NASA/NREN Quality of Service Workshop
Ames Research Center

August 18-19, 1998

NASA Research & Education Network

"Tomorrow's Networking Applications Today"

Contents

1. Call for Position Papers for NASA/NREN QoS Workshop

NASA will host a workshop on QoS for the Next Generation Internet on August 18-19, 1998 at NASA Ames Research Center, Moffett Field, CA. The goals of the Workshop are to share recommended approaches and experiences in the area of Quality of Service in the Next Generation Internet.

Participants will include networking experts from government, industry and academia. Attendance will be by invitation only.

Workshop discussions will be centered around selected position papers. You are invited to submit a position paper on one of the following topics:

QoS application experiences
QoS routing
QoS over ATM
Network analysis/measurement
QoS modeling
QoS policy issues
QoS middleware
Future directions for QoS

Papers, which should be no longer than 1000 words, should be submitted in ASCII text to qos-ws@ciocc.arc.nasa.gov. Important dates are:

Submission of white paper:	June 19, 1998
Notification of acceptance:	July 20, 1998
Workshop:	August 18-19, 1998

For additional information, contact: qos-ws@ciocc.arc.nasa.gov

Or, visit the NREN web site at http://www.nren.nasa.gov.

2. Session I—NASA QoS Application Requirements

Realtime Shuttle Launch Digital Video Distribution

EOS Requirements for Network QoS

2.1 Real-time Shuttle Launch Digital Video Distribution

Mark Foster
NASA Ames Research Center
Mafoster@nren.nasa.gov

Ken Freeman
NASA Ames Research Center
Freeman@nren.nasa.gov

1. Background

The Real-time Shuttle Launch Digital Video Distribution project will provide real-time video of shuttle launch and pre-launch activities to NASA Principal Investigators (PIs) located across the country. Cameras located around two launch pads provide views of the shuttle prior to and during launch. Cameras located in the payload and final assembly rooms provide views of those activities prior to launch. PIs will be able to select from a subset of these video channels in real-time using standard desktop computer platforms. Distribution of the video across a data network infrastructure instead of dedicated leased lines will provide for cost savings, greater flexibility, and potentially improved image quality.

2. Technology Summary

A mixture of technologies will be used for this project. Video will be converted from analog (NTSC composite) to digital (MPEG2, MJPEG) using coder/decoders (codecs) designed for this purpose. The video encoding and compression will permit transmission of multiple streams over the wide area network (WAN). Both IP multicasting and ATM transmission will be used to distribute the video to multiple sites. A graphical user interface will be deployed to allow camera views to be selected in real time by each PI. Since Kennedy Space Center (KSC) is flooded with Web traffic during launch, ATM-based and IP-based Quality of Service technologies will be used to provide for efficient bandwidth utilization.

3. Project Description

This project entails the real-time transmission of a number of digital video streams from the Shuttle pre-launch and launch activities at Kennedy Space Center to other NASA Centers, and ultimately, other non-NASA sites. Because of the number of elements and people involved, we envision multiple project phases, with each phase building on the previous one.

An initial demonstration phase will make use of video digitizers that produce high quality MPEG2 streams over an ATM infrastructure. We will establish ATM circuits between KSC, Ames Research Center (ARC), Johnson Space Center (JSC), and Marshall Space Flight Center (MSFC). A Lucent encoder at KSC will receive NTSC video feeds from one of the video distribution centers, digitize a select set of these using MPEG2 compression, and transmit the resulting streams over the ATM network. Receiving sites will have Lucent decoders for conversion of the MPEG2 stream into viewable images. ATM traffic shaping will be used to support the demands of the video streams during periods of congestion. This phase will help us optimize the performance of the ATM infrastructure.

A second phase will make extensive use of native IP multicasting over the NREN ATM backbone, to reach a broader audience of PIs. Either the Lucent codecs, or similar real-time video digitizing technology will be used; the streams will be encapsulated in IP packets, rather than ATM cells. A preferred class of service will be defined for each multicast group, to improve video frame delay and discard characteristics. This phase will examine the router processing capacities, and better demonstrate the local desktop network needs in conjunction with the wide area capabilities.

There will be additional phases that work to push hard on the edges of the technology, and help to show how reductions in some of the limits can improve the utility to the PIs. Higher resolution images and a greater number of concurrent video streams are two of the goals of this further work.

4. Challenges

We view the key challenges to the project as fitting into three basic categories: organizational, technology, and usability.

Organizational

There are a diverse set of cameras and a diverse set of people viewing the camera signals. Coordination of the digital video distribution will require collaboration among infrastructure support individuals at each of the sites. Institutional support will be needed for some of the costs associated with the effort. While we don't expect any one of these factors to prevent the project from proceeding, to make it a coordinated cohesive project will require some organizational efforts. By undertaking the project in incremental phases, we expect these efforts to be evolutionary, involving more participants as the project progresses.

Technology

The primary technical complexities involve network traffic, infrastructure capabilities, and user interfaces.

The present KSC web-based video snapshots become marginally useful during peak periods, such as a launch. This problem appears to be the result of network congestion at the time of launches. By using ATM classes of service in an ATM-only solution, and by using IP differential classes of service for an IP multicast solution, we expect to alleviate some of the difficulties encountered with the congestion.

The current WAN and LAN infrastructures also bear scrutiny. We expect to deploy OC3 (155 Mbps) service between KSC and sites receiving the video distribution. In some cases, this service will be shared with NISN, so only a portion of the bandwidth (50-60 Mbps) will be available, imposing an upper bound on the number of transmitted streams. In addition to the wide area constraints, getting 6+ Mbps of streamed video to the desktop may be difficult at most sites. The first demonstrations will involve ATM and/or switched ethernet to a very limited set of desktops.

The IP multicasting may put significant burden on some of the routers; while we have increased buffering capacities on most of the routers, there may be performance bottlenecks in the router interfaces that will be uncovered only under heavy load.

Finally, some end-user software development and web interface programming may be required to provide a cohesive interface for the selection and viewing of the video streams. One piece of particular note is the possible re-encapsulation of an ATM MPEG2 stream into an IP multicast stream at full frame rate, for the multiple streams.

Usability

The usability of the video streams is of paramount importance. The packaging, or selection of source streams and resolution(s), is critical to the utility of the distribution. While providing the complete set of analog video sources (over 100) to each viewer at contribution-quality encoding might be desirable, we estimate this would require close to 1 Gbps. Bandwidths available to the desktop using existing technology preclude such a distribution. Thus, we expect to work closely with some of the principle receivers of the

video streams to define which cameras are most important, and what picture qualities (resolution, frame rate) are necessary for useful viewing.

5. Participants

The participants in this project will collectively make it a success. Currently, we expect to have corporate technology and infrastructure support from the following:

- Cisco, Fore, Sprint, and possibly, Qwest for networking
- Lucent, Fore, and possibly other codec vendors for real-time video digitization

In addition to NREN, the following organizations are involved in the support, planning, deployment, and demonstration of the project:

- Ames Research Center – NASA TV broadcasting/multicasting
- Kennedy Space Center – Payload Operations Internet Systems Lab, Operational TV
- Goddard Space Flight Center / NASA Integrated Services Network
- NASA Space Operations and Management Office

6. Related Activities

KSC is in the midst of a complete upgrade of their analog video camera, switching, and distribution system to an all-digital system, capable of much higher performance and resolution (i.e., HDTV). They expect the system to be controllable from designated remote workstations. As both the project described in this paper and the KSC Digital TV project progress, there may be opportunities to demonstrate even broader capabilities than mentioned herein.

We also have not mentioned digital archiving and playback of the selected video streams; in the future, we expect to be able to collaborate with organizations that have applicable expertise and resources in support of such activities.

2.2 EOS Requirements for Network QoS

Andy Germain
NASA Goddard Space Flight Center
Andy.Germain@gsfc.nasa.gov

1. Introduction

EOS has several requirements to support specific flows over networks. These flows are typically repeated daily or more frequently. Some require only modest bandwidths; others are approaching DS-3 rates.

Since these flows are critically important to EOS, the "legacy" implementation was to implement dedicated circuits of the appropriate size between the communicating nodes. Only limited sharing was permitted. For example, multiple dedicated circuits might be mux'ed out of a larger purchased circuit, or an undersubscribed backbone could be used for a portion of the flow. These schemes tend to be inefficient. In the mux case all bandwidth was reserved for specific users, so if any user did not have traffic at a specific instant, that allocation was wasted -- it typically could not be made available to another user, even on another mux channel on the same circuit.

Recently, however, NASA's Integrated Services Network (NISN) has procured a backbone network between several NASA centers and selected other locations, using high performance ATM service from Sprint. In addition to its high speed, this service offers the promise of "managed sharing", in which services can be provided to meet requirements using the common infrastructure, while allowing any excess capacity to be allocated to those users with traffic to send.

If successful, this approach should result in much more efficient operation, and provide improved performance at lower cost. Initially, the savings would be attributed to the economy of scale achieved through consolidation; further savings are expected through implementation of QoS.

This paper will describe the requirements EOS will seek to have met while sharing this ATM infrastructure.

2. EOS Flow Requirements

The following types of EOS Flows can potentially use the ATM backbone:

- EBnet Production
- Instrument Support Terminals
- QA Flows
- Clock and Data
- Voice
- Video Conferencing
- Science Visualizations

The following sections describe these requirements in detail (based on the AM-1 mission where appropriate).

2.1 EBnet Production Flows

Three main types of EBnet production flows can be identified.

First, EOS AM-1 data arrives at EDOS at GSFC after receipt at ground stations. EDOS extracts the instrument data from the telemetry stream in "Level 0" processing. Higher level processing is performed at DAACs, for example, CERES production is performed at LaRC. The level 0 data must therefore flow from EDOS to the appropriate DAACs.

Second, for MODIS (level 1 processing at GSFC), portions of the processing at level 2 and above are performed at the EDC and NSIDC DAACs. So the level 1 MODIS data must be sent from GSFC to these DAACs.

Finally, some processing uses products produced at other DAACs, so these products must also be transferred.

This production process and its required data flows is critical to EOS, so the network used must ensure that the flows are successful in meeting their transfer time requirements.

These flows are normally quite regular, on a daily basis. However, there are various contingencies, which will require additional network capacity, such as recovery from network or system downtime. In addition, it is expected that the production flows for higher level products for the AM-1 mission will "ramp-up" to their full values over a period of up to three years. Following the AM-1 mission are further missions, which will increase the transfer requirements.

Table 2.1-1 shows the planned inter-DAAC EBnet flows for 1999. The values are in kbps, and include an overhead factor of 2.34.

TO / FROM	EDC	GSFC	JPL	LaRC	NSIDC
EDC	-	1301	3	1	200
GSFC	113969	-	2692	32527	9700
JPL	140	1292	-	1	100
LaRC	2748	248	7	-	
NSIDC	200	50	200		-

Table 2.1-1 EBnet 1999 Flows

2.2 Instrument Support Terminals

Instrument Support Terminals are provided to Instrument Team scientists to monitor (in real time) and develop commands for their instruments. IST flows are critical to instrument operations, and it is expected that guaranteed bandwidth would be configured to meet the performance levels required. All IST communications are between the IST and the EOC at GSFC.

The IST requirements are:

Function	Parameter	50 %	90 %	99 %
File Transfer (7 MB)	Transfer Time	3 Min	4 Min	8 Min
Interactive Commands	Round trip Time	0.5 Sec	1 Sec	2 Sec
Real Time (16 kbps)	Latency Jitter	1 sec	2 sec	4 Sec

Figure 2.2-1 shows the IST locations:

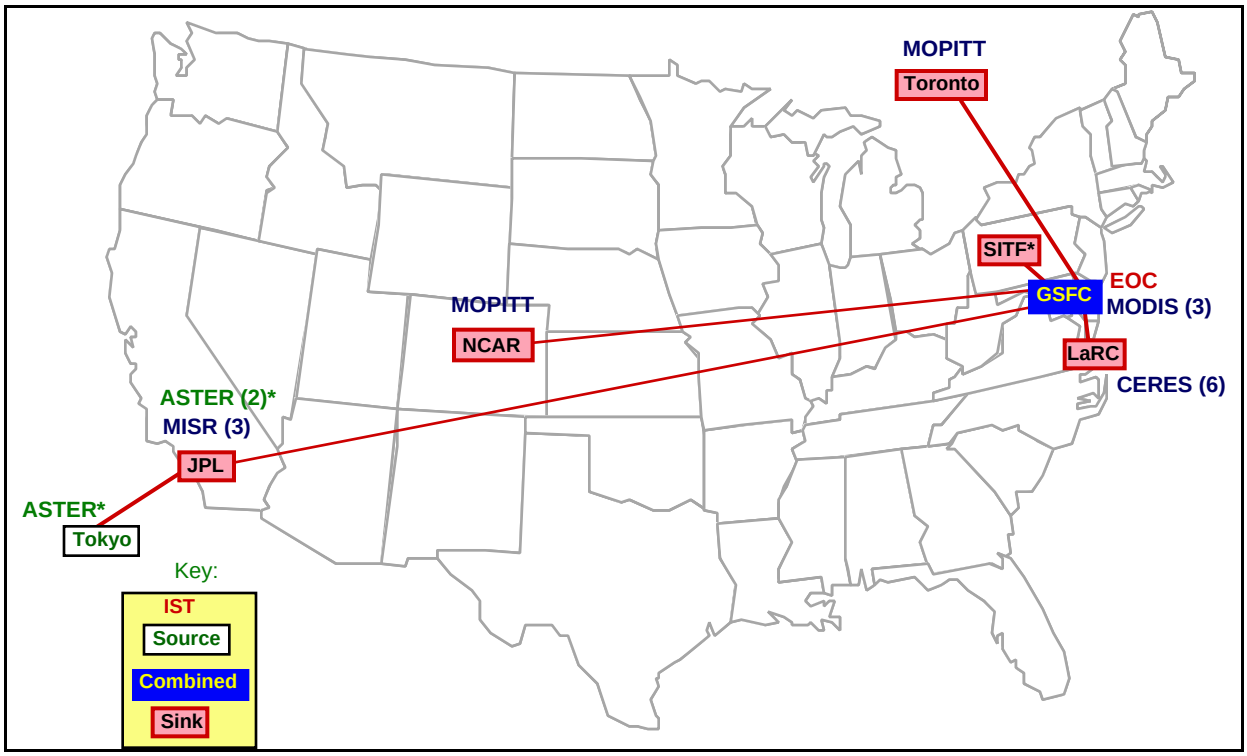


Figure 2.2-1 IST locations

IST sites which can potentially be connected via the Sprint ATM network include

- JPL via NASA Sprint ATM
- NCAR via NASA Sprint ATM peering with vBNS
- Toronto via NASA Sprint ATM peering with CA*NET 2

QoS will be likely be easier to implement within just the NASA Sprint network than when peering is required.

2.3 QA Sites

QA SCFs perform quality evaluations of the production data as part of the instrument teams. This data may be produced at any of the DAAC sites above. Again, the QA flows are high priority, and it is expected that guaranteed bandwidth would be configured to meet the performance levels required.

Figure 2.3-1 shows the QA locations and flow requirements:

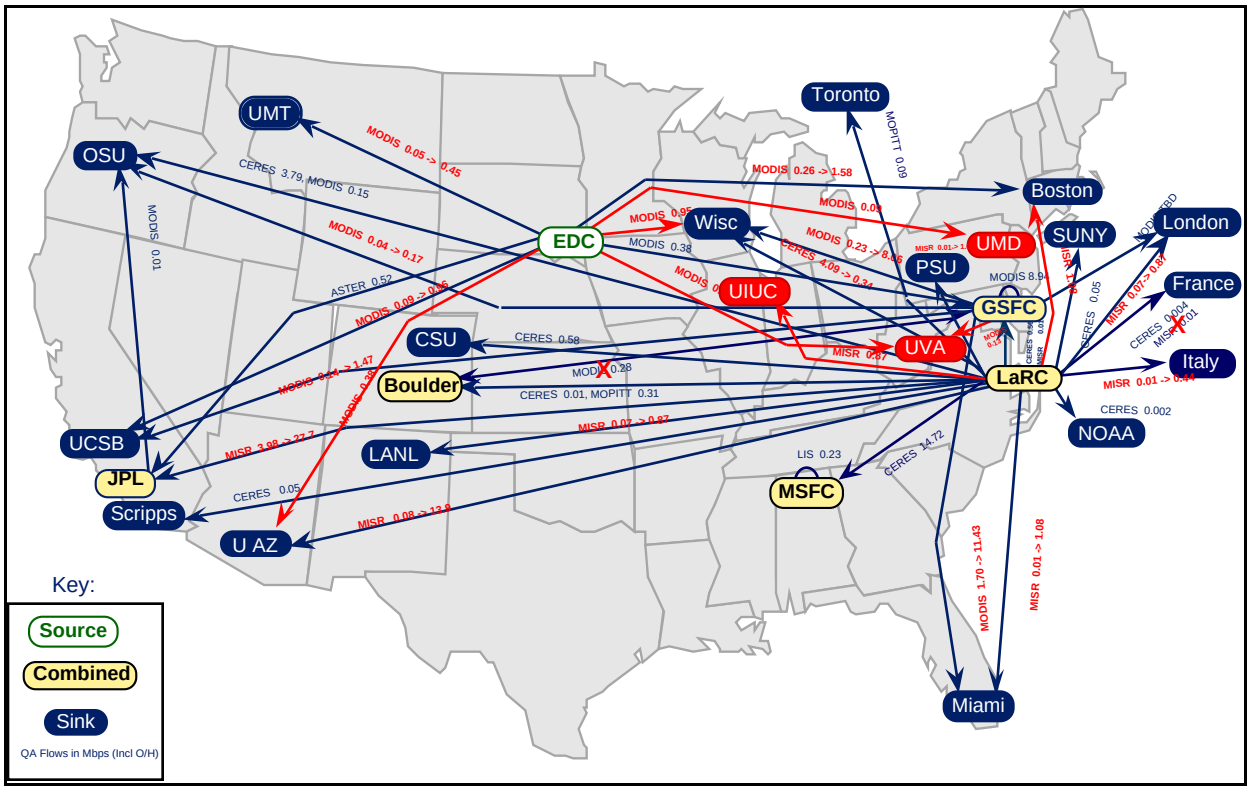


Figure 2.3-1 QA locations and flow requirements

QA sites which can potentially be connected via high performance networks include all DAAC and IST sites above, and in addition:

- via NASA Sprint ATM: MSFC (GHCC)
- via NASA Sprint ATM peering with vBNS: Wisconsin, Boston, Miami, Oregon State, Penn State, U AZ, UCSD, UCSB
- via NASA Sprint ATM peering with ESnet: LANL

Some other QA sites, of which we are uncertain of their connectivity to high performance networks, include Colorado State University, and SUNY-SB.

2.4 Clock and data

Dedicated clock and data circuits are currently used for transporting spacecraft and instrument telemetry data from ground stations -- where it is received from the spacecraft -- to data processing centers. The following types of flows are supported:

Type	Rate (kbps)	Example
Real Time	0.1-32	Housekeeping
Rate Buffered	50 Mbps typ	Science

These flows usually contain the forward error correcting codes used from space to ground, to reduce the effect of transmission errors. On both the space to ground and ground to

ground segments there is no provision for retransmission -- the data is sent with the error correction codes, and the receiving systems must deal with whatever is received. TCP and/or IP are not used.

The likely mapping of these flows onto ATM with QoS would be to CBR PVCs.

Sites involved in these flows for EOS include ground stations in White Sands, NM, Fairbanks AK, and Norway.

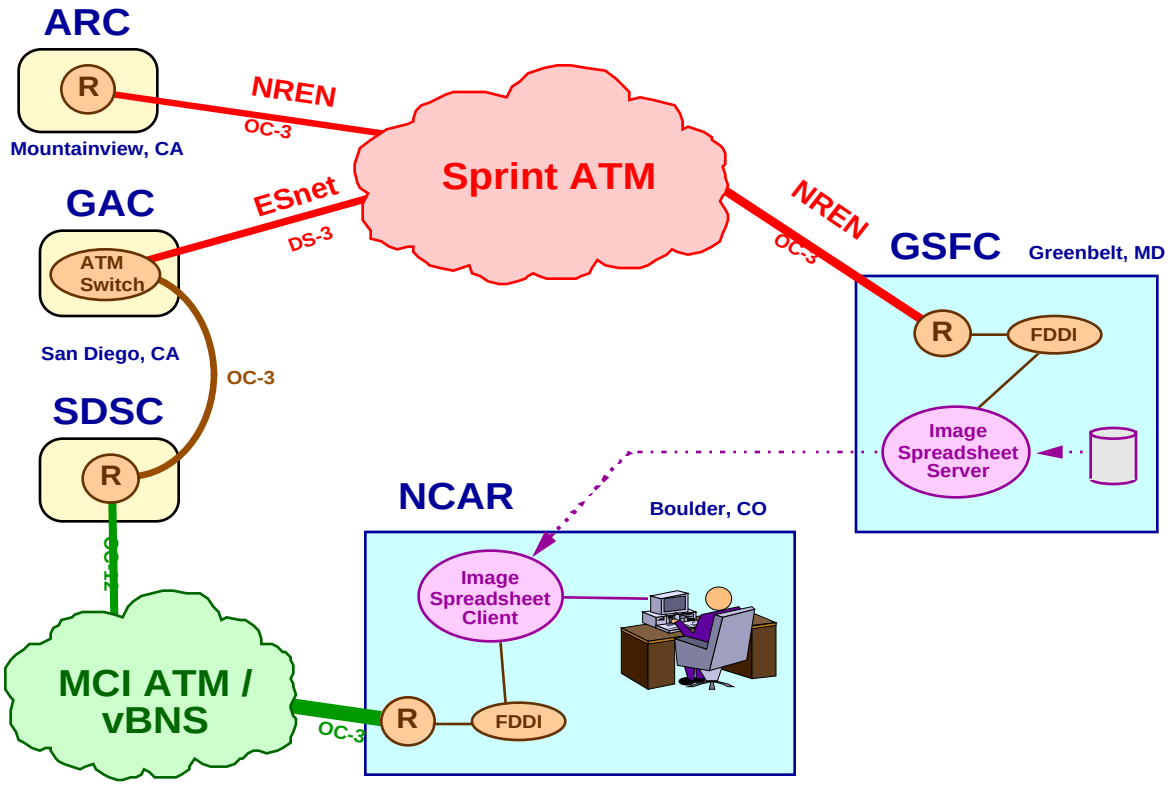
2.5 Voice loops

Voice Loops are used for coordination of Launch and other operations. Accordingly, they are at the launch site (VAFB), IST sites, and the EOS at GSFC. They are currently implemented using dedicated or mux'ed circuits. Requirements are for normal quality voice service

2.6 Scientific Visualizations

In order to use the vast amount of science data produced by EOS, investigators will have to used advanced tools. One class of such tools provides visualizations of time a series of data. One such tool is the Image Spreadsheet, developed by Fritz Hasler at GSFC.
Figure 2.6-1 shows a demo configuration of the image spreadsheet.

Figure 2.6-1 Image SpreadSheet Demo Configuration: 6/97



In order to provide effective visualizations over networks, using remote data, it has been found that bandwidths on the order of 100 mbps are currently required. Work is also underway to reduce these requirements.

3. Session II—QoS Middleware

Bandwidth Reservation: QoS as Middleware

Adaptive QoS Middleware Framework for Complex Flexible Applications

QoS Middleware for the Next Generation Internet

3.1 Bandwidth Reservation: QoS as Middleware

W. E. Johnston
Lawrence Berkeley National Laboratory
wejohnston@lbl.gov

Gary Hoo
Lawrence Berkeley National Laboratory
hoo@george.lbl.gov

1.0 Rationale

Major scientific instrumentation systems, like DOE's synchrotron X-ray sources at LBNL, ANL, and BNL, the gigahertz NMR systems at PNNL, high energy particle accelerators at half a dozen DOE labs (SLAC, Fermi, ORNL, LANL, etc.), are all national user facilities - they involve scientific collaborators throughout the country, and internationally. They are sources of massive amounts of data, generated at high rates (hundreds of megabits/sec, and more), and all of this data requires complex analysis by scientific collaborations at the DOE Labs and at hundreds of universities. E.g., see [1], [3], [4] and [5]. The hundred thousand data tape archive systems needed to organize and preserve the data from such instruments are typically distributed among a few very large facilities that are commonly located at sites other than the instruments, and therefore must be accessed remotely. The intellectual and computing capacity to analyze the data is distributed among all of the sites participating in the scientific collaborations, thus analysis depends on many network based services to aggregate and utilize the required computing and storage components.

Every aspect of this environment is dynamic and requires the ability to move data among geographically dispersed sites at very high rates based on pre-scheduled "contracts" for the diverse resources that make up the data collection, storage, and analysis systems, and the network resources that connect them together.

Supporting the routine creation of robust, high data-rate, distributed applications that support critical, but transient network uses, such as scientific instruments that produce data only during experiments and specific data analysis exercises, necessitates developing various network functionality and middleware services, including the ability to establish, high-speed, end-to-end network bandwidth reservations among the elements of distributed systems.

Bandwidth reservation is one aspect of the general resource reservation problem, but one that is central to the environment, and one that involves unique network issues.

2.0 Integrating Multi-stakeholder Access Control and QoS

We propose a model for bandwidth reservation that can be used in the context of a general resource reservation scheme, but at the same time stay within the scalable model of the differentiated classes of service as described in the IETF diffserv Working Group documents ([6]).

The basic idea is to have bilateral end node agreements that "reserve" bandwidth in the sense that a site actively manages allocation against a class of service. The overall limits on the class of service are established in the service-level agreements between the site and the ISP, but the allocation of flows to this class is closely managed at the site egress.

Further, the resource allocation should be policy based in a way that allows automated reservation, and it should also be possible to proxy one's policy based authority to another site so that the bilateral agreements necessary for inter-site application operation happen automatically. (See, e.g., [2].)

The network level technology to accomplish this is provided by the classifier/shaper/policer functions of the diffserv "traffic conditioner" (TC) element. Layered on top of the TC is a "slot" allocation mechanism ("bandwidth manager"). When instantiated, this slot is a "micro flow" in the diffserv terminology.

Reservation requests are made to the bandwidth manager. The identity of the requestor (user_A), together with the requested resource (time slot, source id, bandwidth) are compared with policy. If the requestor and resource meet the policy, a reservation is made (the slot is allocated and the available bandwidth in the SLA is decremented) and a certificate (a digitally signed document) is issued by the bandwidth manager to represent the reservation.

When, at some point in the future, a request is made to instantiate the flow (i.e. start the instrument or application) the bandwidth manager retrieves the certificate (based on the requestor id and flow characteristics), validates the user and certificate, and instantiates the flow.

The flow characteristics are passed to the TC for classification and enforcement. From the point of view of the ingress router of the ISP, the SLA is never violated because the site bandwidth manager does not over allocate and the TC enforces flow characteristics as reserved.

Another important component in this architecture is a bandwidth broker. This service interacts with the bandwidth manager at the target site in order to accomplish the bilateral reservation. The model here is that some entity ("user_B") at the target site ("site_B") is the (willing) receiver of the flow. The site_B entity must have the right (i.e., be within the policy of site_B) to utilize this flow. User_B conveys (a priori) its authority (in the form of a proxy certificate) to user_A, and the site_A bandwidth broker presents this proxy to the site_B bandwidth manager in order to accomplish the reservation. The site_B incoming flow could probably just be authenticated based on the flow spec matching the reservation (i.e., site_B trusts site_A to authenticate the flow when it is instantiated), although more elaborate authentication is possible.

3.0 References and Notes

Unless otherwise noted, all of these paper are on the Web, and pointers may be found at http://www-itg.lbl.gov/~johnston/papers .

[1] "Real-Time Widely Distributed Instrumentation Systems," William E. Johnston. In The Grid: Blueprint for a New Computing Infrastructure. Edited by Ian Foster and Carl Kesselman. Morgan Kaufmann, Pubs. August 1998.

[2] "Authorization and Attribute Certificates for Widely Distributed Access Control," William Johnston, S. Mudumbai, and M. Thompson. IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises - WETICE'98, Stanford, CA. June, 1998.

[3] "Real-Time Generation and Cataloguing of Large Data-Objects in Widely Distributed Environments," W.Johnston, Jin G., C. Larsen, J. Lee, G. Hoo, M. Thompson, and B. Tierney (LBNL) and J. Terdiman (Kaiser Permanente Division of Research). International Journal of Digital Libraries - Special Issue on "Digital Libraries in Medicine". May, 1998.

[4] "High-Speed Distributed Data Handling for On-Line Instrumentation Systems," William E. Johnston, W. Greiman, G. Hoo, J. Lee, B. Tierney, C. Tull, and D. Olson.

Proceedings of ACM/IEEE SC97: High Performance Networking and Computing. Nov., 1997.

[5] "High-Speed Distributed Data Handling for High-Energy and Nuclear Physics," William E. Johnston, W. Greiman, B. Tierney, A. Shoshani, and C. Tull. Proceedings of Computing in High Energy Physics, Berlin, Germany. April, 1997.

[6] diffserv http://www.ietf.org/html.charters/diffserv-charter.html

3.2 Adaptive QoS Middleware Framework for Complex Flexible Applications

Baochun Li
University of Illinois at Urbana-Champaign
b-li@cs.edu

Klara Nahrstedt
University of Illinois at Urbana-Champaign
Klara@cs.uiuc.edu

1. Background and Motivation

The next generation Internet will provide differentiated services with provision of minimal QoS guarantees to complex and distributed applications. These applications will concurrently share and compete for both end systems resources and transmission bandwidth of heterogeneous multi-protocol networks provided by the next generation Internet infrastructure.

Most of these applications are flexible, in the sense that they can tolerate a QoS range of input quality and resource availability beyond a certain minimum level, and can improve its performance if given a larger share of resources. If resources above the minimum requirements are shared among all applications, statistical multiplexing gain can be improved. In addition, for the flexible applications that involve interactive activities that cannot be predicted beforehand, it may be hard or impossible to specify a maximum demand for QoS.

If these applications specify their QoS requirements with a range representation, adaptation is then desirable to cope with the load and through put fluctuations beyond the minimal QoS guarantees by the next generation Internet. Our goal is to control the behavior of these applications so that they can adapt themselves to fluctuations in resource availability, and their adaptation paths offer graceful degradation facing resource limitations, and graceful upgrades when resources become available again.

2. An Adaptive QoS Middleware Framework

Our objective is to control the adaptation behavior of these applications and fully cope with the dynamics in resource availability over next generation Internet, as well as fluctuations in QoS requirements of the applications themselves. In order to control the behavior of flexible applications, our approach is to embed the control into an adaptive QoS middleware between the OS and applications. This middleware framework interacts closely with Internet II protocol stack (e.g., IPv6) residing in the OS through well-defined interfaces, and monitors the bandwidth allocation to each application in order to make adaptation control decisions.

There are several advantages for placing the adaptation control in the middleware layer: First, it leverages its ability to interact with all applications in the system to ensure fairness and other global properties; second, it is able to enforce different adaptation policies on the applications, based on user's preferences; finally, by using on-the-fly measurements and observation in the middleware framework, the applications can concentrate on major functionalities. In summary, the adaptive QoS middleware framework takes the responsibility of adaptation from flexible applications, which concentrate on processing input and generating output, given a certain amount of resources available.

In our approach, we distinguish clearly adaptation policies from adaptation mechanisms. Adaptation mechanisms are outside of the scope of the middleware. Adaptation policies,

which make choices among mechanisms and decide the degree of adaptation, are determined by the middleware framework. Our approach focuses on the distinctive properties of adaptation policies independent of mechanisms. We propose adaptation policies that are (1) stable so that the adaptation behavior does not oscillate, (2) configurable in terms of adaptation agility to respond fast towards performance disturbances, and (3) fair for all applications so that none of them will starve for resources.

3. Adaptation Tasks in the Middleware Framework

If we examine the interaction between the applications and adaptive QoS middleware framework in a more detailed fashion, it is very natural to map it to a typical control system. In a control system, there is a target system to be controlled. This target system takes appropriate actions to process the input. The input is determined by a controller according to a control algorithm, which monitors the states inside the target system, and compares them to the desired values referred to as the reference. Similarly, adaptation also needs to identify the current states of the target application based on any parameters that can be observed, and to decide input values to the target application in the future.

In order to leverage this analogy between control theory and application adaptation, we use a Task Flow Model [1] to model the structure of the application, with each functional unit modeled as a task. Using this model, we propose Adaptation Tasks that execute adaptation algorithms and control the Target Tasks in the application, as well as Observation Tasks to monitor or estimate the internal states of the Target Task.

Based on the above model, it is possible to apply results in control theory to the design and analysis of adaptation algorithms. The control-theoretical framework allows us to quantitatively analyze system properties such as stability, adaptation agility and equilibrium values for the adaptation behavior. For example, we applied a PID control algorithm in our active adaptation middleware framework and showed the above properties easily [2]. With pre-assigned weights for each application, we were also able to prove that the adaptation algorithm that we have derived satisfied the weighted max-min fairness properties.

4. Distributed Visual Tracking Application: A Testbed

We implemented a testbed based on a complex flexible application, a distributed visual tracking system. The Tracking Server grabs live video and feeds them in real time over a heterogeneous network to the Tracking Client, which performs complex tracking algorithms to track individual objects when they are moving.

The middleware framework has been developed to support the adaptation of this tracking system. These adaptation decisions are decided by the Adaptation Task, executing a PID control algorithm. For the reason that the middleware framework needs to interact with different applications, the interaction goes through a service-enabling platform such as CORBA. In our implementation, the middleware framework and the Visual Tracking application interact with each other through well-defined interfaces written in CORBA IDL. This enables freedom of implementation choices for both middleware and applications. Implemented in Windows NT, the middleware framework runs above the Windows Sockets 2 interface, which encapsulates the Internet protocol stack on the end systems.

Our experiences on implementing the middleware framework to support the adaptation to the complex tracking application have been positive. The output quality, tracking precision, of the tracking algorithms can be kept in a desired range without losing track, when Internet bandwidth availability fluctuates.

5. Conclusions

The objective our adaptive QoS middleware framework is to provide control and direction towards adaptation of complex distributed applications, running over the next generation Internet protocols and services. Adaptation is desired to cope with possible resource fluctuations beyond a minimal QoS level. Based on the adaptation framework that we developed, we are able to quantitatively analyze the stability and adaptation agility of adaptive behavior, leveraging its analogy with control systems. We present experimental results in a distributed Visual Tracking application over Internet networks, in order to demonstrate the effects of adaptation in real systems.

[1] D. Hull, A. Shankar, K.Nahrstedt and J. W.S. Liu, An End-to-End QoS Model and Management Architecture, in Proceedings of IEEE Workshop on Middleware for Distributed Real-time Systems and Services, December 1997.

[2] B. Li, K. Nahrstedt, A Control Theoretical Model for Quality of Service Adaptations, In Proceedings of 1998 Sixth IEEE International Workshop on Quality of Service, pp. 145 - 153. May 1998.

3.3 QoS Middleware for the Next-Generation Internet

Jaroslaw Sydir, Research Engineer
SRI International
sydir@erg.sri.com

Bikash Sabata, Computer Scientist
SRI International
sabata@erg.sri.com

Saurav Chatterjee, Research Engineer
SRI International
saurav@erg.sri.com

In recent years there has been a growing need to run applications that require real-time and other quality of service (QoS) guarantees, over the Internet. However, the lack of real-time services and guarantees in the Internet has limited the large-scale deployment of such applications. These applications have end-to-end QoS requirements that can be expressed in terms of their timing and of the precision and accuracy of the results that they produce. For example, the user of a videoconference application is interested in the quality of the picture (the end result), which is an aggregation of the QoS of capturing the video, compressing it, transmitting it over the network, decompressing it, and displaying it.

Distributed applications use computing, communication, and storage resources. Furthermore, a move towards network computers, appliance computers, and mobile computers is shifting the computing and storage requirements of applications away from end-user machines and towards larger computing and storage servers. In this paradigm, not only communications resources, but also computing and storage servers are shared among many users. To ensure that all of these shared resources are utilized efficiently, and to provide end-to-end QoS to applications, coordinated management of the resources is needed.

We therefore propose that the Next-Generation Internet (NGI) be viewed not as a communication infrastructure comprising a network of networks, but rather as a system of distributed systems. NGI providers should offer not only communication services, but also computing and storage services. Whereas the TCP/IP protocol suite seamlessly connects individual networks into a network of networks, a new middleware layer is required to seamlessly connect individual distributed systems into a system of distributed systems.

The Telecommunications and Distributed Processing Group at SRI International is developing such a middleware layer as part of the End-to-End Resource Management for Distributed Systems (ERDoS) project funded under the DARPA ITO Quorum program [2]. We have developed models of the applications, resources, and system that abstract the implementation-specific details of these entities, allowing our middleware to provide clean interfaces to heterogeneous applications and resources. Our middleware coordinates the activities of applications and resources in performing resource management services such as allocation, scheduling, and QoS-based adaptation of applications. We briefly describe the main features of this middleware in the remainder of this paper.

Interfaces to the Applications and Resources

One of our design goals is to allow the middleware to run over heterogeneous resources and to support heterogeneous applications. We have used a model-driven approach in defining these interfaces. The application model [3] captures the end-to-end structure of an application by dividing the application into its distributed components. The model captures the demands on computing, storage, and communication resources of each application component and the end-to-end QoS requirements and preferences of the application. We have developed software "wrappers" for the various types of applications components. These wrappers implement a common interface to the middleware, allowing the resource manager to control the end-to-end application at run time.

The interface to the resources is based on our resource and system models. The resource model uniformly models the computing, storage, and communication resources and their scheduling attributes. The system model [4] models the system as a collection of distributed systems and their attributes. We have developed resource "agents" that implement a common interface to the middleware, thus hiding the implementation details of specific resources.

QoS Translation, Scheduling, Allocation, Routing, and Graceful QoS Adaptation

We are developing a suite of QoS-driven resource management algorithms, which will be the heart of our middleware. QoS translation is required, to recursively translate top-level application QoS requirements into QoS requirements for lower-level subsystems and resources [5]. Integrated allocation and routing is required, to allocate shared computing, storage and communication resources to each application [6]. End-to-end scheduling over all system resources is required, to guarantee end-to-end QoS [7]. Finally, graceful QoS adaptation is required whenever the system needs to degrade application QoS in response to resource failures or security attacks.

CORBA-Based Implementation

The existence of well-established distributed middleware systems, such as CORBA, makes it desirable to incorporate our middleware into such an established standard. Our goal is to incorporate our middlware into CORBA by incorporating our resource management algorithms into the CORBA Object Request Broker (ORB) and Object Adapter and extending CORBA Interface Description Language (IDL) to capture the information from our application, resource and system models. End-to-end QoS guarantees can then be offered as a CORBA service along with the standard CORBA support for best-effort applications [1].

Conclusion

The design of the NGI will give us a unique opportunity to reevaluate the design of the Internet and propose fundamental changes in the Internet paradigm. We believe that the NGI should provide QoS guarantees to applications that require them (such as voice and video). It is not clear whether such support can be provided by the current TCP/IP based architecture. Also, we believe that the NGI should be viewed as a collection of distributed computing systems instead of as a network of networks, and that a middleware layer should be developed to tie together these systems.

References

[1] Sydir, J., S. Chatterjee, and B. Sabata. 1998. "Providing End-to-End QoS Assurances in a CORBA-Based System," in Proc. International Symposium on Object-Oriented, Real-Time, Distributed Computing (ISORC '98), Kyoto, Japan, pp 53-61 (April).

[2] Sydir, J., S. Chatterjee, and B. Sabata. 1998. (in preparation) "ERDoS QoS Architecture," draft technical report, SRI International, Menlo Park, California.

[3] Chatterjee, S., J. Sydir, B. Sabata and T. Lawrence. 1997. "Modeling applications for adaptive QoS-based resource management," in Proc. High Assurance Systems Engineering Workshop (HASE '97), Washington D.C., pp 194-201, (August).

[4] Sabata, B., S. Chatterjee, J. Sydir, and T. Lawrence. 1998 (in preparation). "Hierarchical Modeling of Systems for QoS-Based Distributed Resource Management," draft technical report, SRI International, Menlo Park, California.

[5] Sabata, B., S. Chatterjee, M. Davis, J. Sydir, and T. Lawrence. 1997. "Taxonomy for QoS Specifications," in Proc. IEEE Computer Society 3rd International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS '97), Newport Beach, California, pp 100-107 (February).

[6] Chatterjee, S. 1997. "A Quality of Service Based Allocation and Routing Algorithm for Distributed, Heterogeneous Real Time Systems," in Proc. 17th IEEE International Conference on Distributed Computing Systems, Baltimore, Maryland (May).

[7] Chatterjee, S., and J. Strosnider. 1995. "Distributed Pipeline Scheduling: A Framework for Distributed, Heterogeneous Real-Time System Design," in Computer Journal (British Computer Society), Vol. 38, No. 4.

4. Session III—QoS Routing

QoS Routing in ATM Networks

Distributed QoS Routing with Imprecise State Information for the NGI

SAAM: A Network Management System for the NGI

4.1 QoS-Routing in ATM Networks

Nina Taft-Plotkin and Bhargav Bellur
SRI International
ninatp@erg.sri.com, bhargav@erg.sri.com

Quality-of-Service (QoS) Routing refers to algorithms that compute paths that satisfy a set of end-to-end QoS requirements. There are a number of complex and interrelated issues involved in the design of QoS routing algorithms and their use in a network. Our work has focused on routing within an ATM network, and thus we only address intradomain issues. In this paper we discuss some of the design issues involved in supporting QoS routing and describe the approaches and decisions we adopt.

In ATM networks, a call can impose requirements on zero to four metrics. These metrics include delay, bandwidth, jitter and loss. Network administrators often want to include a fifth metric, hop count, in the routing process. A call can be admitted on a path if the QoS characterization of the path meets the user's QoS requirements. In order to design practical algorithms, the amount of time spent searching for a path must be limited. If a path cannot be found within that time frame, the call is blocked. In ATM networks, QoS routing algorithms are based on a link-state approach where each node maintains state information on the entire topology. Such information includes the available bandwidth on links, and the delay, jitter and loss guarantees offered across each node or link.

The framework established by PNNI for QoS routing in ATM networks allows for two algorithms to be used to route calls. The first algorithm is for precomputing paths in advance of call arrivals. The second algorithm is used to compute a path on-demand if none of the precomputed paths satisfies the call's QoS requirements. The requirements for these two algorithms differ. The precomputed paths algorithm must find routes from a single source to every destination for a broad range of QoS requirements. The on-demand algorithm only needs to find paths to a single destination since it is executed after the call request has arrived and thus the destination is specified. The requirements on computational speed for the precomputed paths algorithm need not be strict since this algorithm is run in the background. However the requirements on computational speed for the on-demand algorithm are strict since this is run while the call is waiting to be routed.

Network designers have the choice to adopt an approach that uses only one of the two algorithms, or both. We believe that both algorithms should be used. The precomputed paths algorithm can keep the typical call setup time small as long as the vast majority of calls can be routed on precomputed paths. The role of the on-demand algorithm is to keep the blocking rates as low as possible by trying to route calls that could not be satisfied by one of the precomputed paths. This step is important in dynamic networks because the precomputed paths can become outdated. The on-demand algorithm will use the most recent resource status information to compute a path.

It is well known that computing general routes with multiple constraints in a NP-hard problem. However, approximation algorithms are often good enough since we only need to find feasible paths and not optimal ones. For example, we use an on-demand path computation algorithm that is an approximate and abbreviated version of Handler and Zang [1], which presents a dual algorithm for the constrained shortest path problem. We restrict the number of invocations of Dijkstra's algorithm in this method to a small number (e.g., five), in order to keep call setup time low.

Our approach to handle multiple metrics is to spread them across the algorithms and policies used in the routing process. In the precomputed paths we focus the search for paths based on delay and bandwidth. This approach of prioritizing two metrics is useful when metrics are correlated. For example, if delay and jitter are highly correlated than finding a low delay path may also yield a low jitter path. Our on-demand algorithm addresses bandwidth and loss requirements by first pruning unsatisfactory links, and then computing paths that satisfy delay and jitter constraints simultaneously. Since hop count does not explicitly satisfy user requirements, we incorporate hop count into a policy that determines the order in which precomputed paths are tried.

Determining what type of paths to precompute is one of the key design issues in QoS routing in ATM networks. We believe that the computation of alternate paths is very important for QoS routing. During the path setup phase, a particular link or node may reject a call because of insufficient local resources. Hence the next precomputed path tried should be one that avoids the blocking link. Trying to compute completely disjoint paths may not be useful since two totally disjoint paths between two nodes may not exist. We adopt the approach of computing maximally link disjoint paths. Since this approach minimizes the number of common links between two paths, it increases the likelihood that the second path avoids the blocking link.

A second aspect to the problem of selecting which types of paths to precompute involves the service categories. One approach is to precompute separate paths for each of the five ATM service categories. If we precompute multiple paths per service category per destination then the amount of storage needed for these paths creates a scalability problem. We note that the set of metrics to satisfy for one service category is typically a subset of the set of metrics to guarantee for another service category. For example, nrtVBR has requirements on delay and bandwidth, whereas, rtVBR has requirements on delay, bandwidth and jitter. Thus paths computed for rtVBR can also be used for nrtVBR. We prefer this path sharing approach in which precomputed paths can be used by calls from any service category. This latter approach is also appealing because it does not require the network administrator to guess at the fraction of total calls that will belong to each service category.

Another important issue is that of selecting a precomputed path to route an incoming call when several precomputed paths satisfy the requirements of the call. Our general approach is to order the precomputed paths, after they are computed, according to certain policy (e.g., from smallest to largest hop length, or from maximum to minimum bandwidth). Then, upon arrival of a call request, we traverse this list to find a feasible path. Should the first feasible path found be assigned to the incoming call? Or should we assign the path whose QoS guarantees are closest (in a lexicographic sense, implying that there is a certain ordering of the QoS metrics) to the QoS requirements of the call? The advantage of the former approach is that it speeds up call setup time. However the disadvantage is that this approach may raise the overall blocking probability since it does not consider the needs of future calls. If a call request arrives with a large (unstrict) delay requirement, it may be preferable to assign a large delay path in order to save the paths with smaller delay for future calls with stricter requirements. This approach requires one to search through the precomputed paths for the path with the closest fit. The latter approach is advantageous as long as the number of precomputed paths is small.

[1] G. Handler, and I. Zang, "A Dual Algorithm for the Constrained Shortest Path Problem," Networks, Vol. 10 (1980), pp 293-309.

4.2 Distributed Quality-of-Service Routing with Imprecise State Information for the Next Generation Internet

Shigang Chen
University of Illinois at Urbana-Champaign
s-chen5@cs.uiuc.edu

Klara Nahrstedt
University of Illinois at Urbana-Champaign
klara@cs.uiuc.edu

1. INTRODUCTION

The next generation high-speed Internet is expected to support a wide range of communication-intensive, real-time applications. The provision of quality-of-service (QoS) relies on resource reservation. Hence, the future Internet will be connection-orient. The data packets of a QoS connection (flow) are transmitted along the same network path, on which the required resources are reserved.

The diverse QoS requirements of the applications raise new problems. One of the key challenges is QoS routing [3,6,9], whose primary goal is to find a network path that has sufficient resources to meet the QoS requirements of a new connection. In addition, a routing solution should select the low-cost paths whenever possible in order to achieve the global efficiency in resource utilization. The commonly used cost metric is either a hop count or a function of the link utilization.

The QoS routing consists of two basic tasks. The first task is to collect the state information and keep it up-to-date. The second task is to find a satisfactory path for a new connection based on the collected information. As the Internet grows larger and larger, the first task is increasingly difficult [4]. First, the link-state or distance-vector protocols used in the current Internet update the state information at every node periodically or upon triggering when significant state changes are detected. There exists a tradeoff between the update frequency and the overhead involved. For large-scale networks, the excessive communication overhead often makes it impractical for the update frequency to be high enough to cope with the dynamics of network parameters such as bandwidth and delay. Second, the hierarchical approach is likely to be used to solve the scalability problem of large internetworks. However, the state aggregation in hierarchical routing increases the level of imprecision, because it not only loses the detailed state information but also aggregate the existing imprecision. Consequently, the network state kept at any node in the Internet will be inherently imprecise.

In this paper, we propose a ticket-based distributed QoS routing scheme for the next generation Internet. Our scheme works with dynamical network conditions and allows the state information maintained at every node to be imprecise. Extensive simulations showed that the scheme achieves a high success ratio and low-cost routing paths with a modest overhead. It can tolerate a high degree of information imprecision. Readers are referred to [1] for a detailed version of the paper.

2. IMPRECISION MODEL

We propose an imprecision model in this section. Comparing to the probabilistic imprecision model in [5,7], our model is much simpler and can be easily implemented. Let us use the delay metric as an example. The network state maintained at each node i is a table with an entry for every possible destination t. Each entry contains at least two values. One value is the estimated end-to-end delay from i to t, and the other value is the estimated delay variation, i.e., the maximum change of the end-to-end delay before the next state update.

The end-to-end delay is updated periodically by a distance-vector protocol. The delay variation can be easily calculated based on the recent delay history [1].

The upper delay bound is equal to the estimated delay plus the delay variation. The lower delay bound is equal to the estimated delay minus the delay variation. The actual delay from i to t is expected to be between the upper delay bound and the lower delay bound in the next update period. Although it is always possible for the actual delay to be out of this range, such probability can be reduced by choosing either a smaller update period or a larger delay variation.

The imprecision models on other QoS metrics such as bandwidth can be defined in the same way.

3. ROUTING BY TICKET-BASED PROBING

Based on the imprecision model, we design a distributed routing scheme, called the ticket-based probing, which searches multiple paths in parallel for a satisfactory one. We use the delay-constrained routing as an example to illustrate the idea. The QoS routing with other constraints or multiple constraints can be handled similarly. A path that satisfies a given delay bound is called a feasible path.

When a connection request with a delay bound requirement arrives, probes (routing messages) are sent from the source node toward the destination node to search for a low-cost feasible path. Certain number of tickets are issued at the source node according to the current resource availability. Each probe is required to carry at least one ticket. Hence, the maximum number of probes at any time is bounded by the total number of tickets. Since each probe searches a path, the maximum number of paths searched is also bounded by the number of tickets. The proposed routing scheme utilizes the state information at the intermediate nodes to guide the limited tickets along the best paths to the destination, so that the probability of finding a low-cost feasible path is maximized.

In order for the above scheme to work, there are two problems to be solved: (1) how many tickets the source node should issue, and (2) how to propagate the tickets in the network in order to find a low-cost feasible path.

The solution to the first problem relies directly on the imprecision model. If the delay requirement is too small to be possibly satisfied, i.e., smaller than the lower delay bound, then no tickets are issued and the connection is rejected. If the delay requirement is larger than the upper delay bound and thus can be surely satisfied, the minimum number (>= 1) of tickets is issued. Otherwise, if the delay requirement is between the lower bound and the upper bound, multiple tickets are issued with more tickets for smaller delay requirements [1].

There can be many different types of tickets with different purposes. In [1], two types of tickets are defined: yellow tickets and green tickets. The purpose of yellow tickets is to maximize the probability of finding a feasible path. Hence, yellow tickets (more precisely, probes carrying them) prefer paths with smaller delays, so that the chance of satisfying a given delay requirement is higher. The purpose of green tickets is to maximize the probability of finding a low-cost path. Green tickets prefer the paths with smaller costs, which may however have larger delays and hence have less chance to satisfy the delay requirement. Other types of tickets may be introduced, whose propagation is based on a combination of delay and cost.

When an intermediate node receives a probe, it splits the probe to one or more new probes, distributes the received tickets among these new probes and forwards them to a selected

subset of neighboring nodes [1]. There are three general rules governing the propagation of probes.

a. Each probe must contain at least one ticket. Hence, the number of tickets in a received probe upper bounds the number of new probes that can be created.

b. Different types of tickets are distributed among the new probes differently. For example, a probe sent toward the direction with a smaller estimated delay to the destination should have more yellow tickets, and a probe sent toward the direction with a smaller estimated cost to the destination should have more green tickets.

c. The state information of the intermediate nodes is collectively used to guide the probes along the best paths toward the destination [1]. In addition, a probe proceeds only when the delay of its path is no more than the delay requirement. Hence, once a probe reaches the destination, it detects a delay-constrained path.

Details about the scheme can be found in [1]. Simulation results show that, when the state information is imprecise, the ticket-based probing approach performs much better than the traditional shortest-path approach in terms of success ratio and average path cost. The average message overhead is modest and controllable, which makes it scale well. Hence, this routing scheme is suitable for the next generation Internet.

4. NICE PROPERTIES

In the following, we outline a number of nice properties of the ticket-based probing scheme.

First, the routing overhead is controlled by the number of tickets issued, which allows the dynamic tradeoff between the overhead and the routing performance.

Second, the proposed scheme is designed to work with imprecise state information. The level of imprecision (information uncertainty) has a direct impact on the number of tickets issued. Multi-path parallel search increases the chance of finding a feasible path and thus provides a means to tolerate information imprecision.

Third, our scheme considers not only the QoS requirement but also the optimality of the selected path. Low-cost paths are given preference in order to improve the overall network performance.

Fourth, a distributed routing process is used to avoid any centralized path computation that could be very expensive for QoS routing in large networks. The state information kept at the intermediate nodes are collectively used to find the best path.

Fifth, it can work nicely with some existing connection establishment protocols [8] and resource reservation protocols [10].

5. ON-GOING WORK

Multimedia applications tend to have diverse QoS requirements on bandwidth, delay, delay jitter, cost, path length, etc. From a network designer's point of view, it would be beneficial to accommodate different QoS routing algorithms in a single integrated framework, which captures the common messaging and computational structure. The framework should be simple, which enables an efficient implementation, and extensible, so that new QoS metrics can be easily added without affecting the existing ones. It should also support both unicast and multicast. To the best of our knowledge, all existing algorithms

are tailored towards specific problem classes with a single or multiple specific routing constraints, and none of them provides a framework with the above features.

We have been working on an integrated QoS routing framework based on the imprecision model for high-speed packet-switching networks [2]. The framework is fully distributed. By using the ticket-based probing, the framework provides a uniform way to utilize the available imprecise information. It allows the tradeoff between the overhead and the routing performance. Different distributed routing algorithms (DRAs) can be quickly developed by specifying only a few well-defined constraint-dependent parameters within the framework. Three families of DRAs can be derived:

(1) unicast DRAs using only local states,

(2) unicast DRAs using imprecise global states, and

(3) multicast DRAs built on top of the unicast DRAs.

Extensive simulation shows that the overhead of the proposed algorithms is stable and modest. Overall, this unique QoS routing framework contributes to the network support of heterogeneous and distributed multimedia applications with diverse end-to-end QoS requirements.

REFERENCES

[1] S. Chen and K. Nahrstedt, Distributed QoS Routing with Imprecise State Information, Technical Report, University of Illinois at Urbana-Champaign, Department of Computer Science, 1998, available upon request, s-chen5@cs.uiuc.edu

[2] S. Chen and K. Nahrstedt, Distributed Quality-of-Service Routing in High-Speed Networks Based on Selective Probing, Technical Report, University of Illinois at Urbana-Champaign, Department of Computer Science, 1998, available upon request, s-chen5@cs.uiuc.edu

[3] S. Chen and K. Nahrstedt, On Finding Multi-Constrained Paths, IEEE International Conference on Communications, June, 1998

[4] S. Chen and K. Nahrstedt, An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions, Technical Report, University of Illinois at Urbana-Champaign, Department of Computer Science, 1998, available upon request, s-chen5@cs.uiuc.edu

[5] R. Guerin and A. Orda, QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms, Infocom'97, Japan, April, 1997

[6] C. Hou, Routing Virtual Circuits with Timing Requirements in Virtual Path Based ATM Networks, INFOCOM'96, 1996

[7] D. H. Lorenz and A. Orda, QoS Routing in Networks with Uncertain Parameters, Infocom'98, March, 1998

[8] D. Ferrari and D. C. Verma, A Scheme for Real-Time Channel Establishment in Wide-Area Networks, IEEE JSAC, April, 1990

[9] Z. Wang and J. Crowcroft, QoS Routing for Supporting Resource Reservation, JSAC, September, 1996

[10] R. Braden (Ed.), L. Zhang, S. Berson, S. Herzog, and S. Jamin, Resource reSerVation Protocol (RSVP) Version 1, Functional Specification (draft-ietf-rsvp-spec-13.ps) INTERNET-DRAFT, Internet Engineering Task Force - RSVP WG, July, 1996

4.3 SAAM: A Network Management System for the NGI

Geoffrey G. Xie
Naval Postgraduate School
xie@cs.nps.navy.mil

1. Motivation

One major objective of the NGI is to support all types of data using a single network. This integrated services requirement poses a significant new challenge to network management: namely, Quality of Service (QoS) path management. Specifically in the NGI, the capacity of each link will be shared by a set of logical service pipes, each of which provides a particular level of packet performance measured by a set of QoS parameters [2,6]. Typical QoS parameters include the bound on packet delay and the rate of packet loss. For a data flow (e.g., a video flow) that requires end-to-end QoS guarantees to its packets, the source will invoke a resource reservation protocol such as RSVP [1] to establish a QoS path to the destination. The path is composed of a sequence of service pipes whose composite QoS meets the flow's requirement. In summary, in addition to maintaining connectivity, the NGI must dynamically allocate and maintain QoS paths.

QoS path management is not an easy task, especially for the NGI, which will be a dynamically changing environment where hardware faults, service malfunctions, and user misbehaviors frequently occur. QoS parameters, especially those for real-time data (e.g., interactive audio), are often measured at a granularity of milliseconds. Moreover, when faults do occur, the management system should reconfigure QoS paths automatically and efficiently, before the user notices any problems. Therefore, a dominating requirement for a network management system for the NGI is that it must be proactive, i.e., able to detect and react to changing network conditions within fractions of a second. More specifically, the system should interact with the reservation protocol and maintain useful information about paths so that a replacement path can be established quickly when a currently used path becomes no longer useful. Existing network management systems cannot meet this requirement. In these systems, such as in the SNMP architecture [3], each router maintains a local Management Information Base (MIB) of its state. Typically a human administrator (or a software tool) at a management station queries routers to learn about the current state of a path when a need arises, e.g., when a drastic degradation of performance is reported by users, or when new hardware or software is added. There is no automated mechanism that periodically aggregates data from routers into "ready to use" performance information about paths. The critical MIB data is transported using UDP or TCP, which do not guarantee timely delivery. Therefore, existing management systems are reactive in nature. They are not designed to automatically detect and reconfigure within a short time frame in the face of network faults and service malfunctions. They also do not interact with the reservation protocol. When a QoS path is no longer useful, the application that uses it would have to re-invoke the reservation protocol to establish a replacement path, and as such, the application's performance would suffer.

Another problem with using current network management systems for the NGI is that these systems would cause too much processing at routers. First of all, QoS (i.e., path based) routing algorithms must deal with more constraints, and thus require much more processing on the part of each router [9]. Moreover, it has been shown that it is desirable to use different QoS routing algorithms under different conditions to improve network performance [10]. Having such flexibility also requires more computation at each router. Therefore, processing overhead will become a major concern if every router is required to perform QoS routing and re-routing. The problem is compounded by the fact that to

support integrated services, a router will have to use a packet forwarding method that is much more elaborate than FIFO.

2. SAAM Architecture

We at the Naval Postgraduate School are developing a Server and Agent based Active Management (SAAM) system for the NGI. The SAAM project is currently sponsored by DARPA under the NGI initiative. To explain the intuitions behind our approach, consider road traffic monitoring and control during commute hours in a large city such as San Francisco. In this case, radio stations are the main management entities. They send out helicopters to monitor traffic on roads in their respective coverage region. The information from the helicopters is aggregated at the stations and advice is then broadcast in real-time to commuters. The advantages of using a helicopter include (1) allowing monitoring of traffic over long routes ("paths"); such monitoring is required to produce advice such as "It will take about 20 minutes to go from Bay Bridge to Civic Center following I-80", and (2) enabling early detection of congestion, which is key for effective control. (In contrast, each individual motorist can only monitor traffic within a short radius and cannot independently foresee congestion.)

While current network management systems behave like road traffic monitoring that depends mostly on reports from individual motorists, our SAAM architecture follows the helicopter model. SAAM achieves timely network management by employing a set of management servers ("helicopters") that we will refer as SAAM servers. These SAAM servers and associated mobile agents will periodically collect state information about the network and maintain "ready to use" path performance data in a Path Information Base (PIB). For scalability, the SAAM servers will be organized in a hierarchy [7]. At the lowest level, each server will maintain a PIB for paths within a small network region. Upper level servers will handle performance data for paths that cross multiple regions. Essentially, the SAAM servers form a logical overlay network between the management station and the physical network. As a result, the routers are relieved of most routing and network management tasks [7,8].

The SAAM servers and the routers will use a real-time transport protocol for data and agent communications. Therefore in SAAM, most management and control tasks will be performed by the servers in an automated and timely fashion. Only a small number of planning tasks will require human interaction, and in such cases, the management station will need to communicate with a high level SAAM server most of the time. SAAM will also have built-in mechanisms to interact with the reservation protocol and provide it useful path information when requested.

3. Related Work

The use of route servers has been proposed for data networks before [9]. The motivation was to reduce the computational overhead for a set of closely associated routes, e.g., a set of Internet Service Provider (ISP) routers that share a Network Access Point (NAP). QoS routing and re-routing were not considered. The ATM PNNI standard has proposed to use a hierarchical mechanism for path based routing (i.e., set-up of VPs and VCs). SAAM differs from PNNI in two main areas. First, SAAM will support any QoS routing algorithm with a comprehensive PIB [8] while PNNI has been designed specifically for an Open Shortest Path First (OSPF) routing algorithm. Second, SAAM will rely on dedicated servers for routing and other network functions while PNNI requires sophisticated processing at switches. We envision that a SAAM server is nothing more than a PC running SAAM specific software. As such, SAAM will be able to support faster deployment of new services than is currently possible.

REFERENCES

[1] Zhang, L., et al, "RSVP, A New Resource ReSerVation Protocol," IEEE Network Magazine, pp. 8-18, September 1993.

[2] Floyd, S., and Jacobson, V., "Link-sharing and resource management methods for packet networks," IEEE/ACM Transactions on Networking, 3(4):365-386, August 1995.

[3] Case J., and et al, "The Simple Network Management Protocol," Internet Draft, RFC 1157, May 1990.

[4] Yu, J., Manning, B., and Rekhter, Y., "Router server technical overview," Technical report, January 1998, HTML document http://www.rsng.net/overview.html.

[5] ATM Forum, "Private Network-Network Interface (PNNI) specification," version 1.0, March 1996.

[6] Xie, G.G., and Lam, S.S., "Real-time block transfer under a link sharing hierarchy," IEEE/ACM Transactions on Networking, 6(1):30--41, February 1998.

[7] Xie, G.G., et al, "SAAM: An integrated network architecture for integrated services," in Proceedings of 6th IEEE/IFIP International Workshop on Quality of Service, Napa, CA, May 1998.

[8] Xie, G.G., et al, "Efficient management of integrated services using a path information base," submitted for publication, available at http://www.cs.nps.navy.mil/people/faculty/xie/pub.html.

[9] Wang, Z., and Crowcroft, J., "Quality of service routing for supporting multimedia applications," IEEE Journal on Selected Areas in Communications, (7):1228--1234, September 1996.

[10] Ma, Q., and Steenkiste, P., "On path selection for traffic with bandwidth guarantees," in Proceedings of 5th IEEE International Conference on Network Protocols, pages 191--202, Atlanta, GA, October 1997.

5. Session IV—Internet2

Internet2 QoS and Differentiated Services

5.1 Internet2 QoS

Ben Teitelbaum,
Internet2 / Advanced Network & Services
Ben@advanced.org

Internet2 is a collaborative effort by over 120 U.S. universities, in cooperation with industry and government, to develop and enable the advanced networked applications that universities need to fulfill their research and education missions into the next decade. To provide the needed quality-of-service (QoS) functionality, the Internet2 QoS Working Group has recommended testbed trials of the evolving differentiated services (diffserv) approach to QoS. Meeting the needs of the most demanding advanced applications will require particular emphasis on services that provide "absolute" transmission assurances to end-to-end flows.

Goals and Requirements

The primary goal of Internet2 is to support the research and education missions of universities through the development of new advanced networked applications, example components of which may include:

* Two-way interactive audio/video
* Collaborative virtual environments
* High-fidelity streaming video and audio
* Remote instrument control
* Large data transfers
* Unknown future technologies

Unfortunately, many valuable networked applications are not feasible in today's best-efforts internet because they require certain minimum end-to-end performance assurances.

Application QoS requirements may be characterized by a set of transmission parameters and corresponding assurances for a specified traffic profile. Example transmission parameters are loss, latency, and jitter. Traffic profiles are usually specified by token bucket filters, bounding bandwidth and burstiness.

Dialogue between the Internet2 applications and engineering communities has revealed a general understanding of which transmission parameters matter most (loss and latency), but very little understanding of the assurances that are required. Closer scrutiny reveals a broad range of needs.

The most demanding applications have specific (absolute) requirements grounded in hard thresholds of human perceptual sensitivity and are highly intolerant of network performance below these levels. Other applications have specific requirements, but are tolerant of occasional drops in performance or require only that the performance averaged over a certain time period be maintained. Finally, there is a desire for network functionality that can simply treat traffic from certain applications or users better than other traffic. In the last case, the assurance made is relative to the network load imposed by other traffic, while in the first two, the assurances are absolute.

Whereas QoS focuses mostly on the nature of the service "floor", it is also important to characterize the "ceiling". Over the past 25 years, considerable expertise has been developed around the engineering of so-called "adaptive applications". These applications are able to adjust gracefully and usefully to the varying availability of network resources. In the presence of QoS, adaptive applications are able to exploit network resources beyond their base needs. There is significant concern that any Internet2 QoS approach allows for continued use of adaptive techniques.

Additional application needs include the ability to schedule advanced reservations (for applications like distance learning) and the extension of QoS to multicast flows. The Internet2 QoS Working Group understands both to be important goals, but not immediate requirements. Other key design requirements identified by the working group are similar to those of the Internet at large and include:

* Interoperability of network equipment and clouds
* Scalability
* Administratability
* Measurability
* Support from operating systems and middleware
* Incremental deployment starting in 1998

The most challenging of these technical requirements are interoperability and scalability. It is crucial that any design allows for concatenating the services of multiple, separately administered and engineered clouds into meaningful end-to-end services. Furthermore, any QoS approach considered must scale to the large numbers of flows and high packet-per-second rates found in core routers.

Differentiated Services

In the past year, there has been an increasing interest in simple and scalable approaches to IP QoS. Within the IETF, this interest has resulted in the creation of the Differentiated Services Working Group. Diffserv reduces the state requirements of core routers by carefully aggregating QoS-enabled flows into aggregates that are given a small number of simple differentiated forwarding treatments indicated by bit settings in the packet headers. A broad and flexible range of services is provided by protecting access to the aggregate treatments with per-flow policing at the network periphery.

The differentiated treatments (dubbed "per-hop behaviors" or "PHBs") suggest, but do not imply, particular queuing disciplines and consequent services. Proposed examples include: default (best-efforts) forwarding, expedited forwarding ("forward me first"), and drop preference ("drop me last"). Each diffserv flow is policed and marked at the first trusted router according to a contracted service profile. Downstream from this leaf router, flows are aggregated and all subsequent forwarding and policing is performed on aggregates.

An important benefit of handling traffic aggregates is to simplify the construction of end-to-end services from the concatenation of multiple cloud-to-cloud services. Individual network clouds (administrative domains) contract with neighboring clouds to provide differentiated service contracts for different traffic aggregates. Like the per-flow contracts, aggregate contracts are characterized by profiles, which are enforced at cloud-cloud boundaries. This model results in a set of simple bilateral service agreements that mimics current interprovider exchange agreements.

Finally, to make appropriate internal and external admissions control decisions and to configure leaf and edge device policers correctly, each cloud is outfitted with a bandwidth broker. A host signals its local bandwidth broker to initiate a connection and the user is authenticated and subject to local policy-based admissions control decisions and resource accounting. Then, on behalf of the requesting user, the local bandwidth broker initiates an end-to-end call setup along the chain of bandwidth brokers representing the clouds to be traversed by the application flow. The bandwidth broker abstraction is critically important because it allows separately administered network clouds (possibly implemented with very different underlying technologies and polices) to manage their network resources as they see fit.

Diffserv Evaluated

The objectives that are guiding the evolution of diffserv are remarkably similar to the requirements and design goals for Internet2 QoS. In particular, the QoS working group finds attractive diffserv's emphasis on simplicity, scalability, interoperability, and administratability. Also attractive is the range of services that appear to be supportable under the framework.

We conjecture that a set of four particular proposed diffserv services spans the space of Internet2 application requirements:

* Premium - emulates a leased line; peak bandwidth guarantee with low delay and jitter (intolerant applications)

* Assured (or a similar) - emulates a lightly loaded network; similar to Controlled Load (tolerant, adaptive applications)

* Class-of-service (CoS) - relative, precedence-based service classes; better best-efforts service to meet coarse user and institutional priorities
* Default - best-efforts service

Diffserv is also attractive for the administrative flexibility that it would afford to member campuses. In the diffserv framework, each cloud is free to set its own arbitrarily complex and baroque local policies as long as the bilateral agreements that it makes with other clouds are honored. Some campuses may elect to avoid the complexities of highly dynamical admissions control and policy enforcement by selling statically configured, subscription-based services to their users.

Finally, there is happy coincidence in the concurrent evolution of diffserv and Internet2 QoS -- while the push for differentiated services is being driven by the immediate needs of commercial network service providers to offer CoS, the set of new functional components that are needed overlaps significantly with those that are required to implement the experimental, absolute, per-flow services described above (Premium and Assured). Traffic shapers, classifiers, policers, and much of the proposed bandwidth broker functionality either exists today or will in next-generation routers.

Nevertheless, many hard technical and social problems remain that can be solved only through empirical experience and iterative design refinement in a testbed environment. In the coming months, Internet2 will facilitate a diffserv testbed where new services can improve incrementally in quality and availability and where technical experience can be gained for feedback to the relevant research, engineering, and standards efforts. On the social side, it is particularly important to begin to develop broader experience with QoS, so that developers, users, and administrators can begin to understand the expectational and policy transitions that are required before production QoS services can flourish.

6. Session V—QoS Testing & Modeling

Analysis of a Hierarchical, Link-Sharing, Network Traffic Control Implementation on TCP and UDP Data Flows

Wide Area QoS Testing Experiences, Expectation vs. Reality

NetSim^{Q}: A Java-Integrated Network Simulation Tool for QoS Control in Point-to-Point High Speed Networks

ARMing NREN's Advanced Applications—Measuring End-to-End WorkFlow QoS Requirements

6.1 Analysis of a Hierarchical, Link-Sharing, Network Traffic Control Implementation on TCP and UDP Data Flows

George Uhl
NASA Goddard Space Flight Center
Uhl@mamba-e.gsfc.nasa.gov

1.0 Introduction

The Earth Observing System Data and Information System (EOSDIS) is projected to be one of the largest users of non-defense networks in the world with data traffic rates of about nearly two terabytes per day by the year 2004[1]. Likewise data retrieval rates are expected to be high. EBnet and NSI Wide-Area Networks (WANs) will provide internal and external network services to the EOS community. Over-provisioning resources to meet demand could be cost prohibitive and result in inefficient utilization of network bandwidth. A more resource efficient, less costly approach is to apply traffic control mechanisms on EOSDIS networks to maximize the utilization of bandwidth across multiple applications without having to procure additional resources to meet the requirements of a particular application. Emerging network quality of service (QoS) mechanisms should be able to meet the goals of efficiency at a reasonable cost for EOSDIS networks.

At the network layer, QoS mechanisms are still in their infancy and primarily within the domain of the research community. The FreeBSD operating system is a popular version of Unix for network research since the operating system and supporting utility source code is freely distributed with each release and it has well-established, robust networking capability. One of the first implementations of QoS in FreeBSD has been Kenjiro Cho's Alternate Queueing (ALTQ) prototype[2]. Initially released in March 1997 as a FreeBSD kernel upgrade from a traditional first-in-first-out (FIFO) queuing kernel to a Class-Based Queuing (CBQ) traffic control manager, ALTQ has undergone continued enhancement and development to become a stable research and prototyping tool. CBQ is a hierarchical, link-sharing, network traffic control discipline which supports the separation of traffic into hierarchical classes and applies bandwidth allocations to each class[3]. Every class is derived from a parent class and can share some or all of its parents' allocated bandwidth. The ESDIS Network Prototyping Lab chose ALTQ CBQ as a representative implementation of a traffic control manager suitable to the high volume, low delay requirements of EOSDIS.

2.0 Approach

Using a FreeBSD kernel configured with ALTQ for traffic management, TCP and UDP data flows were processed over an ATM OC3 link. Behavior of the data flows was observed using FIFO queues and CBQ queues. Because most of the anticipated data flows over EOSDIS networks will use TCP for transport; this study focused on the effects of concurrent UDP and TCP data flows on a single designated TCP data flow.

3.0 Testbed Configuration

Three PCs were configured with FreeBSD with two designated as hosts and the other as a router as shown in Figure 1. The router's kernel was upgraded with release 1.0.1 of the ALTQ software. Host A and the Router were data flow sources with Host B acting as a data flow sink. The hosts were connected to the Router using ATM OC3 permanent virtual circuits (PVCs) configured in unspecified bit rate (UBR) mode. CBQ, when applied, was enabled on the Router interface that connected the Router with Host B.

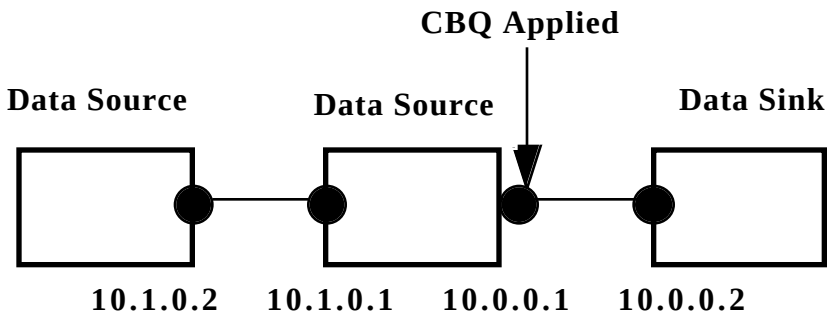


Figure 1. QoS Testbed Configuration

4.0 Test Scenarios

Data flows were generated from Host A and the Router and arrived at Host B. On tests involving single streams Host A was the data source. On tests involving two data streams, both Host A and the Router were data sources. The TTCP network performance benchmark tool was used to generate and receive data streams and measure throughput and latency. The FreeBSD network analysis tool NETSTAT was used to measure errors.

Seven scenarios were designed to measure the effectiveness of CBQ traffic management over various combinations of TCP and UDP data streams. The first two scenarios establish a baseline performance for TCP and UDP data streams in the testbed. Scenario 3 measured how concurrent UDP and a TCP streams behave using traditional FIFO queue processing, while Scenarios 4 and 5 measure how CBQ managed both streams with bandwidth preference given first to TCP and then to UDP. Scenario 6 measured how two concurrent TCP streams behave using FIFO queue processing, while Scenario 7 measured how CBQ controls the TCP streams giving preference to one of the steams.

Five tests were performed for each scenario.

5.0 Test Results

The ALTQ software managed data flows very close to the parameters specified in the CBQ configuration file for each scenario. The table below averages the results of the tests performed for each scenario.

Test Results Summary Table:

Scenario	Queuing Discipline	CBQ Rate	Average Throughput (Mbps)			Average Latency (msecs)		
			TCP-A	UDP	TCP-R	TCP-A	UDP	TCP-R
1	FIFO	-	127.24	-	-	0.50	-	-
2	FIFO	-	-	128.12	-	-	0.50	-
3	FIFO	-	13.72	127.88	-	4.91	0.50	-
4	CBQ	75% TCP	92.02	-	-	0.69	-	-
		20% UDP	-	24.64	-	-	2.60	-
5	CBQ	20% TCP	23.86	-	-	2.68	-	-
		75% UDP	-	91.24	-	-	0.70	-
6	FIFO	-	65.71	-	58.73	0.99	-	1.09
7	CBQ	75% TCP-A	90.21	-	-	0.71	-	-
		20% TCP-R	-	-	24.3	-	-	2.63

A=Host A
R=Host A w/Router

No input or output errors were detected, nor were any TCP packets dropped while testing.
6.0 Observations

When using a FIFO queue, the rate at which TCP packets were output to the queue was determined by the rate at which queued packets could be delivered. The UDP data flow, being unimpeded by flow control constraints, overwhelmed a concurrent TCP data flow even when that TCP data flow had already been established. When using CBQ, the UDP data flow was constrained to the specified rate reserved for it over the controlling interface permitting the TCP data flow to achieve steady-state.

When using a FIFO queue with two concurrent TCP data flows, the data flows oscillated as TCP congestion control procedures were invoked. With CBQ, both TCP data flows are constrained to their specified bandwidth limits and the TCP flow control parameters of reach flow steady-state. The results indicate that sophisticated queuing disciplines can bring order to chaotic data flows.

The TTCP measurements of the UDP data flows were obtained at the data sink. The TTCP data source generated more data (650 to 700 Mbs) than could be output over the OC3 link. TTCP nor NETSTAT account for dropped or lost UDP packets due to the connectionless nature of the protocol. The effects of overflowing output buffers on UDP data flows could be observed with other host-based applications.

7.0 Conclusions

Even though small in scale and simple in nature, these experiments demonstrate the potential of traffic management disciplines within networks. The low cost, high data volume, low transport delay requirements of EOSDIS could be met with networks configured with similar traffic managing disciplines as CBQ. Further analysis will include more data flows from different classes of applications (multicast, real-time) and experimentation using Resource Reservation Protocol (RSVP).

References

1. Lisotta, T. and Radwin M., EOSDIS Technogoly Assessment Study Briefing. 1998. http://spsosun.gsfc.nasa.gov/ETAS_WAN.pdf
2. Cho, Kenjiro. Alternate Queueing for FreeBSD. http://www.csl.sony.co.jp/person/kjc/programs.html
3. Cho, Kenjiro. A Framework for Alternate Queueing: Towards Traffic
4. Management by PC_UNIX Based Routers. In Proceedings of USENIX 1998
5. Annual Technical Conference, New Orleans LA, June 1998.

6.2 Wide Area QoS Testing Experiences

Expectations vs. Realities

Michele Mascari
NASAGSFC / Lockheed Martin
michele.mascari@gsfc.nasa.gov

Tino Sciuto
NASA GSFC / CSC
agatino.sciuto@gsfc.nasa.gov

Dharmesh Shah
NASA GSFC / CSC
Dharmesh.shah@gsfc.nasa.gov

Introduction

This paper summarizes Quality of Service (QoS) over ATM test results. These tests were performed by NASA Integrated Services Network (NISN) in collaboration with Sprint and NASA Research and Educational Network (NREN). These tests were performed to test whether ATM QoS promises can **meet user requirements effectively**. Indeed, we learn that expectations and reality don't match.

Motivation

NISN performed network simulation tests for two potential NASA specific user traffic requirements. They are:

Earth Observing System (EOS) Requirements [1]

1. Transfer several gigabytes of data between NASA sites within one day of receipt at earth stations
2. Minimize the network costs by implementing a managed statistical multiplexing solution
3. Guarantee delivery of data for internal traffic requirements (see expectations below) over external traffic during network congestion

High Rate Serial Data Distribution (HRSDD) Requirements

1. Transmit non-standard[1] format serial data transparently, at rates ranging from a few kbps to 75 Mbps
2. Guarantee delivery in worst-case network congestion
3. Minimize the network costs

Expectations

EOS Expectations

The EOS expectations are:

1. Prioritization - EOS has two kinds of network requirements at EOS Remote Observing System Data Center (EROS Data Center, or EDC), Sioux Falls, SD - internal and external. The internal network requirements have higher priority over the external requirements because they account for data processing and archiving. These tests were to show the capability of the ATM backbone to provide EOS guaranteed delivery of internal requirements at the EOS specified data rate.
2. Minimum bandwidth guarantee during worst-case network congestion
3. Ability to use all available bandwidth when there is no network congestion

HRSDD Expectations

[1] Non-standard means that it is not a telecommunications industry standard format. However, the format follows NASA's standard.

NASA's serial clock and data stream can interface with the ATM network via an ATM Conversion Device (ACD). The ACD stores the serial clock and data in a buffer. While this buffer is being filled, data is sent to a transmit buffer. The ACD converts data from the transmit buffer, taking a block at a time, into cells and transmits them over ATM. Hence, it is inherently bursty. Thus, the HRSDD expectations are:

1. Prioritization - HRSDD traffic is for customers like International Space Station (ISS) and the Space Shuttle Program, which includes mission-critical traffic. Thus, these tests, too, were to show that the ATM backbone could guarantee delivery of this traffic in the worst network conditions.
2. Burst Tolerance - This testing was designed with legacy and future systems in mind and involved the transmission of high rate synchronous data over an asynchronous distribution medium. These tests were to also show that the ACD could interface with the ATM network.

Given the above motivation and requirements, NISN collaborated with Sprint and NREN to simulate operational conditions over the wide area ATM network. We expected that ATM's inherent prioritization order of servicing QoS contracts would guarantee network resources to the above two traffic types during network congestion. ATM switches service network traffic in the priority order Constant Bit Rate (CBR), Variable Bit Rate (VBR), Available Bit Rate (ABR), and Unspecified Bit Rate (UBR) contract. These are specified using UPC (Usage Parameter Control) to specify the desired QoS parameters.

By specifying a suitable VBR contract for EOS requirements, we expected to be able to:

1. Recommend VBR contract values for internal and external traffic requirements
2. Identify the SCR values for internal and external traffic sources that would guarantee a minimum bandwidth and avoid bandwidth starvation during congestion. The SCR for internal traffic would ensure that all internal EOS data would be transferred within a day.
3. Demonstrate that EOS data could burst above SCR and use all available bandwidth

Similarly, by specifying a suitable CBR contract for HRSDD requirements, we expected to be able to:

1. Demonstrate that mission-critical traffic would be guaranteed resources even during congestion
2. Identify the network's capability to accommodate the ACD's burst requirements

EOS Test Configuration

Two workstations, one each at Goddard Space Flight Center (GSFC) and EDC, were configured to simulate the external traffic. Similarly, two other workstations at these sites were configured to simulate internal traffic requirements. ATM PVCs, with different contracts, connected these workstations over the ATM service. The test configuration is shown in Figure 1 below. The contracts were only enforced on portions of network where contention was expected, which is indicated by arrows drawn under the appropriate ATM switch(es) in the the diagram below. The numbers above the arrows correspond to test numbers and also test results tabulated in Table 1. The direction of the arrows indicates the direction of the traffic-flow.

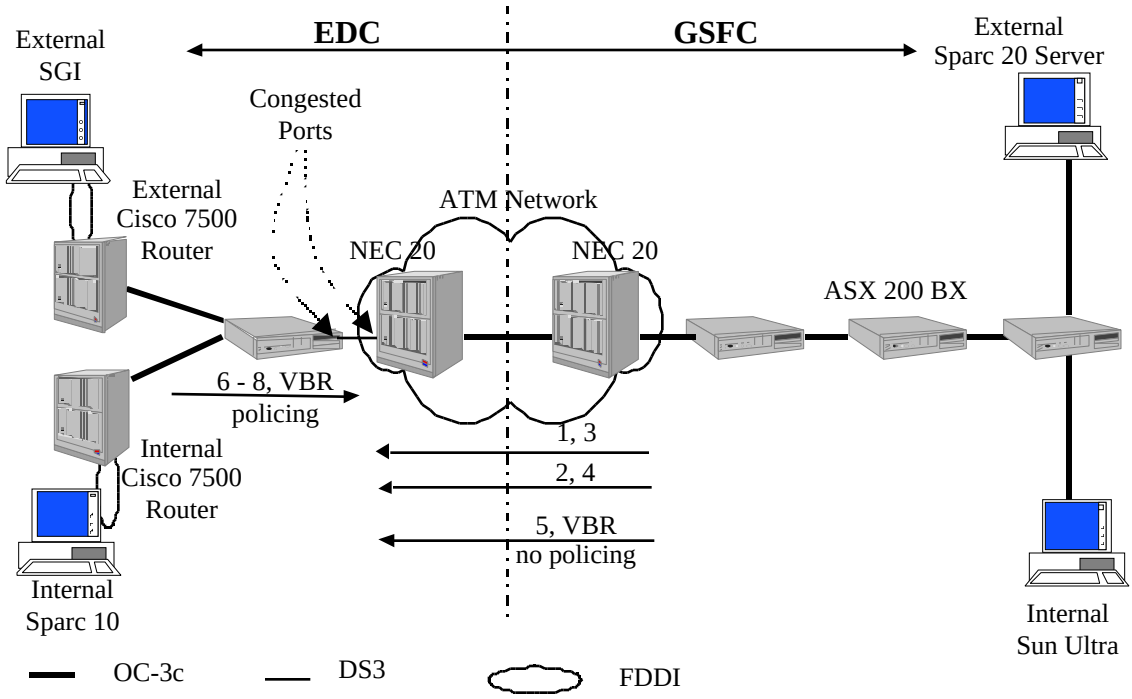


Figure 1 - EOS Test Configuration [2]

The test results are summarized in Table 1 below, where all rates are in Mbps. Tests 1 through 4 are baseline tests and basically show that the throughput is limited to the DS3 link capacity, after accounting for the various overheads. Test results 5 through 8 are "interesting". These results are discussed next.

#	UPC Contract		Expected		Observed		Comments
	Ext.	Int.	Ext.	Int.	Ext.	Int.	
1	N/A	VBR	0	36	0	34.4	OK
2	N/A	UBR	0	36	0	33.9	OK
3	VBR	N/A	36	0	34.7	N/A	OK
4	UBR	N/A	36	0	34.3	N/A	OK
5	No pol	No pol	18	18	17.36	17.38	OK
6	VBR*	VBR*	13.4	23.4	8.23	16.68	High SCR => high throughput
7	VBR*	N/A	13.4	0	9.38	0	Throughput < SCR
8	N/A	VBR*	0	23.4	0	19.26	Throughput < SCR

* Denotes new UPC parameters. External traffic's SCR was 13.4 Mbps, internal traffic's SCR was 23.4 Mbps (after taking away ATM cell header overhead), with Maximum Burst Size (MBS) = 256 cells

Table 1 - EOS Test Results

Test Results Interpretation

Tests 1 through 5

In these tests, VBR contracts were enforced on the "core" ATM network, which comprises the NEC 20 ATM switches. The obtained throughput indicates that the VBR contract on the OC3 portion of the ATM network is available equally to all of NASA traffic on that VBR virtual path. It is important to note that the current policy on all NASA traffic in the Sprint cloud is to not enforce policing. In other words, all cells are serviced regardless of the ATM contract. This approach has served NASA well so far. It allows all NASA sites to have fully meshed topologies and burst capabilities at line rate. This explains the observed throughput in these test runs.

Note that in tests 1 through 5, the flow of traffic was from GSFC to EDC. In contrast, for tests 6 - 8, traffic flowed from EDC to GSFC.

Test 6

In this case, VBR contracts were enforced on the FORE ASX 200BX switch at EDC to test the impact of bandwidth contention caused when both internal and external sources sought to use the same DS3 connection at EDC. The observed results imply the following.

1. Implementing the VBR contract on the FORE switch limits the throughput for both the sources.
2. The switch allocates more bandwidth to the source with higher SCR. But, the aggregate throughput is about 25 Mbps, resulting in significant underutilization of the link capacity, which is 36 Mbps.
3. A minimum bandwidth can be guaranteed to a source, even during congestion, by specifying a higher SCR for its VBR contract.

Tests 7 and 8

These were baseline tests. In these tests, only a single source (internal or external) transmitted traffic, and the available throughput was recorded. The observed throughput indicates that the maximum bandwidth available to either source was slightly less than the SCR specified for the respective VBR contract.

Results 6 - 8 indicate that the VBR contracts were very stringent. Indeed, the CDVT values on these PVCs were 250 μs, which is very stringent. As a result, the TCP traffic was strictly limited to the SCR. The CDVT could not be increased for these tests.

HRSDD Test Configuration

The purpose of the HRSDD CBR tests was to demonstrate the capability to transmit a stream of ATM cells that has a varying data rate component across the Sprint ATM cloud using a CBR contract. The advantage in using a WAN CBR contract is that CBR cells have the highest priority and are less likely to be dropped during congestion (see expectations above). This capability could enable NASA to provide encoded mission video and mission-critical clock and data services using an ATM service, which has the potential to reduce costs as compared to current practices.

CBR Test Configuration

The wide area configuration for CBR tests is depicted in Figure 2 below. A PVC was configured from a Nascom Internetworking Lab Environment (NILE) FORE switch at GSFC, to the Sprint ATM service, via two other FORE ATM switches at GSFC that are not shown in the diagram below. The network switched the PVC to ARC and terminated it

at the NREN FORE switch at ARC. Finally, at NREN, ARC, the PVC was looped back over the same path to the NILE FORE switch at GSFC. CBR contracts were enforced on all switches in the network and at the CPE. Baseline tests were also performed locally within NILE.

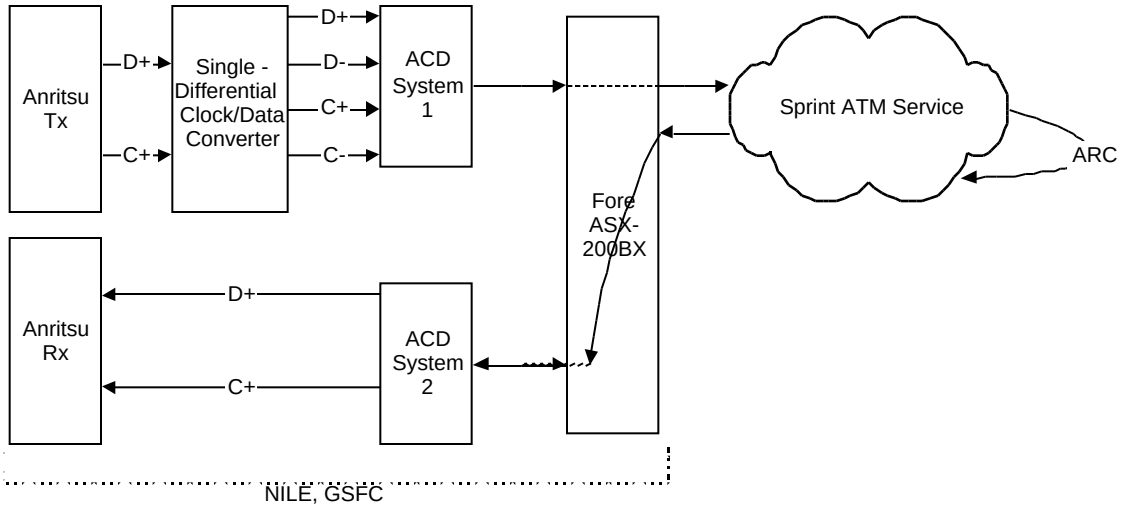


Figure 2 - CBR over ATM Test Configuration [3]

Observed Results and Interpretation

Baseline Tests within NILE

Using ForeThought, we graphed the average cell rate, over 10 second intervals, as observed on the FORE switch. The observed waveform is of a sawtooth pattern, with the variation between the peaks and valleys of about 1,000 cells/second. Note that, at OC3c rates, this is equivalent to a CDVT of approximately 30 ms.

Within the NILE, cells were successfully transferred through FORE ASX-200BX switches using a CBR contract with the CDVT set to 64 ms. In earlier tests, a VBR contract was used and the required CDVT for loss-less transmission was also observed to be 64 ms.

CBR over WAN Tests

We transmitted an average rate of ~6.9 megabits per second (Mbps) in to the network [6 Mbps serial data + ATM Adaptation Layer 5 (AAL5) overhead + ATM cell overhead]. The receiving Anritsu at GSFC recorded only ~2 Mbps. Tests were not performed at higher data rates due to bandwidth constraints on the network connection between GSFC and ARC.

Although Sprint indicated they were passing all cells received within their network, we suspect that the FORE switches at the edge of the network were discarding cells. During the CBR tests in the NILE, we noted that the FORE switches discard cells with no accounting whenever CDVT is violated. During the WAN tests, the CBR contracts configured by Sprint had a CDVT ranging from 250 μs to 271 μs. Prior discussions between other team members and Sprint indicated that for both CBR and VBR contracts, the CDVT is kept at these low levels. Thus, these tests revealed that the key issue in transporting mission-critical data is to have configured the appropriate CDVT values.

Reality

1. Inducing priority by applying QoS contract limits the sources' bandwidth to the provisioned QoS, i.e.; sources cannot use the available bandwidth when the network is not congested.
2. CBR and VBR contracts are enforced very stringently, if the CDVT values are tight. For VBR contracts, this results in underutilizing link capacity, as observed in the EOS tests.
3. Contracts implemented only in the core network area don't allow for differentiated service at the edges.
4. QoS contracts must be implemented end-to-end, and again, this implies that the source and destination must be restricted to the contract values for optimum performance.
5. Though economical, statistical multiplexing using ATM may prove expensive if mission-critical traffic is lost.

Conclusions

The expectations that ATM would introduce "managed" statistical multiplexing and provide optimum QoS is unrealistic. ATM QoS comes with a price. These tests reveal that in order to have end-to-end guarantees, all involved parties need to collaborate. This translates into traffic shaping and limiting the burst capability. The ATM contract specification for VBR is that cells need to comply with SCR, PCR, MBS and CDVT parameters. In order to achieve VBR compliance on a TCP system it is necessary to implement rate-control at the traffic sources, thus creating a rigid end-to-end network framework for data transfer. Thus, the customer is left wondering whether she/he really does have an OC3c connection.

Note that EOS requirements loosely match *prioritized* ABR or UBR service. Similarly, the HRSDD requirements match *prioritized VBR*. The expectation that ATM's implicit prioritization of servicing CBR, VBR, ABR, and UBR contracts would prioritize user traffic is misplaced. CBR, VBR, ABR, and UBR are contractual obligations, and are valid only if the user traffic meets the contracted obligations. The expected traffic does not map very well to ATM's currently defined contract profiles. Besides, currently, ATM does not support prioritization. The **expectations don't map to the reality.**

The results, however, are beneficial to NISN customers, Sprint and NREN because they highlight the network configuration issues that must be dealt with to best meet customer expectations

Acknowledgements

The authors and NISN duly appreciate the efforts put in by Sprint and NREN for facilitating and supporting these tests. Specifically, the authors acknowledge contributions from Leslie Gaillard, Terry Bernard (both Sprint), David Guevara (NREN, ARC, NASA) and Vishy Narayan (Raytheon STX, under contract to NREN, ARC, NASA).

References

[1] Germain, Andy, "EOS QoS Requirements ", August 1998

[2] Sciuto, Tino, "EOS Test Report", July 1998

[3] Mascari, Michele, "ATM Conversion Device (ACD) Constant Bit Rate Test Summary", July 1998

6.3 NetSim^{Q}: A Java-Integrated Network Simulation Tool for QoS Control in Point to Point High Speed Networks

Please see: *http://eewww.eng.ohio-state.edu/drcl/grants/middleware97/netsimQ.html*

6.4 ARMing NREN's Advanced Applications— Measuring End-To-End WorkFlow QoS Requirements

Gary Ramah
NASA Ames Research Center
gramah@mail.arc.nasa.gov

The purpose of this paper is to investigate the Application Response Measurement (ARM) proposal and evaluate its applicability within NASA's Education and Research Network (NREN). ARM provides a way to monitor an application's communication resource utilization.

In June, 1996, Hewlett-Packard and Tivoli Systems announced a collaboration to develop an open, vendor-neutral approach to manage the performance of distributed applications. The Application Response Measurement (ARM) API, is an application programming interface for measuring end-to-end application response time.

ARM works by embedding simple calls within an application's code allowing the application to pass vital information about its internal state to an outside agent. All the application has to do is call the ARM code just before a transaction (or a subtransaction) starts and then again just after it ends.

The primary indicator of application performance, as perceived by the end user, is responsiveness. Network managers, however, usually only get information about an application's responsiveness when an end user calls to complain about the lack of it. This is because network managers monitor network transactions and not user transactions. While they usually have access to performance measures of all the elements in the path of an application flow, they rarely have an accurate picture of the responsiveness user applications achieve.

Essentially, there are two ways of monitoring application responsiveness. One way is to try to piece together information from network tools and other sources. Remote monitoring probes and other tools can be used to examine traffic, server performance can be tracked, database alerts can be monitored and individual applications can provide general. A better way is to insert application hooks that can later be used to provide information on how the application is running. This better approach is difficult to implement because most applications are currently developed without consideration of their manageability in distributed environments.

Application management was never a problem in centralized host environments because mainframes had embedded monitoring facilities that could easily address issues involving application health, response time and administration. But now, applications are being deployed across multiple platforms and administrators are realizing that not being able to monitor or benchmark remote applications leaves them very vulnerable to being blindsided by problems that are hard to diagnose and even harder to solve. Flow analysis attempts to overcome the myopic view by aggregating traffic into composite flows. (http://www.nlanr.net/Flowsresearch)

Flow analysis (packet train model) requires the network monitor to filter packets it observes based on the Data source, the Data sink and some time out interval. If a packet is seen within a specified time window and belongs to the same Data source/sink then it is marked

as a member of the existing flow, if the window is exceeded a new flow is identified. This approach places the flow identification burden on the network monitoring filter instead of on the flow originator itself. This Traditional approach to application flows also relies on static addressing and service requirements. Future advanced application flow's can not be bound by these static requirements and may need to support dynamic addressing and dynamic service requirements.

Advanced applications will be more complex, taking different execution paths and spawning different subtransactions, depending on the results of previous events. Every permutation could take a different form when it goes across the communication link, making it that much harder to reliably correlate network transactions with what the user sees. A user transaction may spawn several other subtransactions, some of which may execute locally and some remotely. Any probes that exist only in the network layer will not see the entire picture. Traditional Flows can be identified by the information contained within each packet/cell either on an end-to-end, link-by-link, or network-by-network basis. Advanced flows can only be identified by the application itself.

Because of the complex logic that will be contained within these advanced applications, the only acceptable solution is for the application itself to participate in the flow identification. This can be accomplished by instructing the application to call the ARM API at the start and end of each transaction. The network monitor can now measure the same transactions that the user sees. There is no guessing about what a transaction is, there are no dependencies on specific protocols, and unusual situations are handled as well as the most common situations.

Using ARM, the application designer can easily mark sections of their application by invoking API function calls at the beginning and end of each important unit of work. ARM is fairly trivial to implement using the six Application Program Interfaces (API's) that can define and mark units of work effectively timestamping an application's trail of activities.

Arm_init - Id's app/User and initializes measurement system
Arm_getid - registers a transaction by name
Arm_start - Indicates the start of a unit of work
Arm_update - shows progress and status
Arm_stop - Indicates the end of a work unit
Arm_end - Disables measurement environment

The application sends these ARM API calls to an ARM client agent (residing on the same machine) which in turn summarizes the data, compares the times with threshold values and possibly sends the summarized data and alerts to the ARM server agent.

Network traffic is steadily increasing at a faster rate then network resources. Currently Network managers react to end user's problems. There is NO CONTROL over network resources, whatever happens … happens! Network managers currently monitor and troubleshoot key application-layer traffic within the enterprise network using RMON2. ARM supplements the RMON2 specification for remotely monitoring application-layer traffic, by enabling administrators and application developers to monitor network applications from within the application itself, acting essentially like a debugger for communication tasks.

ARM is a starting point for research into Network aware adaptive applications. Once networks begin to share information with the application layer, true proactive network management becomes possible. Instrumenting the applications of the future will provide a

method for characterizing how these applications use the various resources of the information infrastructure!

References:

Service Management Using the Application Response Measurement - API Without Application Source Code Modification by Martin Haworth for Hewlett-Packard Company (http://cmg.org/regions/cmgarmw/shortarm.html)

Managing the Enterprise with the Application Response Measurement API (ARM) by Denise Morris for Hewlett-Packard Company (http://hpcc923.external.hp.com/openview/rpm/papers/armwp.htm)

The Application Response Measurement (ARM) API, Version 2 by Mark W. Johnson for Tivoli Systems (http://www.cmg.org/regions/cmgarmw/marcarm.html)

Application Performance Vital Signs by Amy K. Larsen for Data Communications. (http://saxophone.agora.com/roundups/vital.html)

Successful Deployment of IT Service Management in the Distributed Enterprise by Doug McBride, Hewlett-Packard Company (http://www.hp.com/openview/rpm/papers/pmotcd1.htm)

7. Session VI—QoS Policy

QoS Routing in IP Networks: Myths and Realities

A Scalable Resource Management Framework for Differentiated Services Internet

Preliminary Results Evaluating Diffserv Services

Designing for Public Networks

7.1 QoS Routing in IP Networks: Myths and Realities

George Apostolopoulos
University of Maryland
georgeap@watson.ibm.com

Roch Guerin
IBM T. J. Watson Research Center
Guerin@watson.ibm.com

Sanjay Kamat
IBM T. J. Watson Research Center
sanjay@watson.ibm.com

Satish K. Tripathi
Bourns College of Engineering
University of California

Abstract

There is much uncertainty regarding the "real" benefits of QoS routing in IP networks, and our goal here is to shed some (partial) light on this issue. In particular we will try to demonstrate that QoS routing not only offers performance benefits, but is also not as expensive as one could fear. We base this conclusion on evidences we obtained on the processing cost and the protocol overhead associated with QoS routing, which we found to be both containable and well within the capabilities of modern technology. These evidences were gathered based on both detailed simulations and an actual implementation of QoS routing based on an extension to the OSPF protocol [1] that allows selection of routes based on QoS (bandwidth) requirements. In addition to reporting these results, we also provide some insight into various trade-offs available to QoS routing, if its cost, in particular protocol overhead, needs to be further minimized.

1 Background

Because of its potential benefits, Quality of Service (QoS) routing has recently received substantial attention in the context of its possible use in an integrated services IP network. QoS routing is the process of selecting the path to be used by the packets of a flow based on its QoS requirements, e.g., bandwidth or delay. Several recent research results [2, 3, 4, 5, 6] have pointed out the potential of QoS have pointed out the potential of QoS routing for improving network utilization and the service levels provided to requests with QoS requirements. The improvement to the service received by users is in the form of an increased likelihood of finding a path that meets their QoS requirements. Conversely, the improvement to network efficiency is usually in terms of increase in "revenue", where revenue is typically a function of the number of flows or the amount of bandwidth carried by the network.

Despite these benefits, there remains much uncertainty regarding the value and feasibility of implementing QoS routing protocols in IP networks. This is primarily because of the additional costs that support for QoS routing entails. These added costs have two major components: "computational cost" and "protocol overhead". The former is due to the more sophisticated and more frequent path selection computations, while the latter is caused by the need to distribute updates on the state of network resources that are of relevance to path selection, e.g., available link bandwidth. Such updates translate into additional network traffic and processing, in particular in the case of "link state" protocols, on which most of the QoS routing proposals currently being put forward are based, e.g., see [7] for an overview. In such a context, it is important to properly assess the magnitude of those costs, so that the weight of the associated benefits can be better evaluated. Such an assessment

should strive to both understand fundamental cost components, and provide insight into actual implementations and the trade-offs they involve.

2 On the Costs, Benefits, and Trade-Offs of QoS Routing

Our first step was aimed at gaining some basic understanding into the trade-offs that QoS routing involves. For that purpose, we performed detailed simulations to evaluate how routing performance degrades as we vary its protocol overhead by limiting the number of link updates it can send. The main control knob we relied on is the update trigger policy, that is responsible for determining when and how often the link metrics updates on which QoS routing relies, are to be flooded into the network. We experimented with different types of trigger policies, e.g., threshold-based, class-based, and timer-based policies, and identified different operating regions for QoS routing. In particular, we measured the amount of update traffic generated by each policy, and assessed it against the gains in routing performance that it yields. Based on these experiments, we find that it is possible to obtain reasonable performance improvements in routing, even with rather inaccurate link metrics information, i.e., with minimal protocol overhead.

While this first set of experiments did provide some insight into the available trade-offs between protocol overhead and routing performance, it did not give a complete understanding and sizing of the many cost parameters of QoS routing. This is something that can only be done accurately through an actual implementation. For that purpose, we carried out an implementation of an extension to the OSPF protocol, based on the proposal of [8], that supports computation of QoS routes for flows with bandwidth requirements. We then used this implementation, to obtain realistic estimates for the cost of various QoS routing operations such as path computation, generation and reception of QoS link state advertisements (LSA), and compare the overall cost of our QoS enhanced version of OSPF to that of the standard OSPF protocol. The main conclusion of these experiments is that although QoS routing extensions indeed correspond to an increase in overall cost, their impact is minimal given the processing capacity of modern processors.

In addition to stand-alone experiments that provided the individual costs of the different operations required by QoS routing, we also combined this experimental data with simulations in order to better assess actual operational costs in a full-scale network. These experiments confirmed our earlier findings regarding the relatively small relative cost increment introduced by QoS routing. In addition, they indicated that the bandwidth consumed by QoS LSAs was only a minute fraction of the link bandwidth of most networks.

3 Summary

Based on the evidences we have gathered using both an actual implementation and detailed simulations, we believe that QoS routing in IP networks is not only feasible but also cost effective. Especially so as the traffic generated by flows with specific QoS requirements grows in volume. This being said, there are still a number of issues that remain to be addressed to improve the functionality or performance of QoS routing. One of them is support for explicit routes as it provides a better control of the paths used by flows with QoS requirements. Another, probably more important, enhancement is to extend support for QoS routing in OSPF to more than a single area, i.e., through the backbone network and to remote areas. This could then be further extended to support inter-domain area as discussed in [7].

References

[1] J. Moy, OSPF Version 2, Internet Request for Comments, RFC 2178, July 1997.

[2] H. Ahmadi, J. S. -C. Chen, and R. Gu'erin, "Dynamic Routing and Call Control in High-Speed Integrated Networks." In Proceedings Workshop Sys. Eng. Traf. Eng., ITC'13, 1991.

[3] Z. Wang, and J. Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications." IEEE Journal Selected Areas in Communications, 14(7):1228-1234, 1996.

[4] W. C. Lee, M. G. Hluchyj, and P. A. Humblet, "Routing Subject to Quality of Service Constraints in Integrated Communication Networks." IEEE Networks, pages 46-55, July/August 1995.

[5] R. Widyonon, "The Design and Evaluation of Routing Algorithms for real-time Channels." Technical Report TR-94-024, University of California at Berkeley, June 1994.

[6] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Two Distributed Algorithms for the Constrained Steiner Tree Problem." In Proceedings 2nd International Conference on Computer Communication and Networking, pages 343-349, 1993.

[7] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A Framework for QoS-based Routing in the Internet." Internet Draft, QoSR Working Group, work in progress, April 1998.

[8] R. Gu'erin, S. Kamat, A. Orda, T. Przygienda, and D. Williams, "QoS Routing Mechanisms and OSPF Extensions." Internet Draft, work in progress, January 1998.

7.2 A Scalable Resource Management Framework for Differentiated Services Internet

Andreas Terzis, Lan Wang, Lixia Zhang
University of California - Los Angeles
terzis, lanw, lixia@cs.ucla.edu

1. Introduction

The ultimate goal of network QoS support is to provide users and applications with high quality data delivery services. From a router's view point, however, QoS support is made of three basic steps: defining packet treatment classes, allocating adequate amount of resources for each class, and sorting all incoming packets to their corresponding classes. The first step involves standardization, while both the second and third steps require protocol mechanisms that can scale well with the ever increasing network size and speed.

Through joint effort of research community and industry the IETF Differentiated Services (diff-serv) Working Group is reaching agreement on an initial set of definitions for "per-hop behavior" (PHB), a set of differentiated packet treatments at each router based on the TOS field value (called "code-point") carried in the IP header. This work addresses both the first and third issues above: it defines the traffic classes as well as provides a simple packet classification mechanism -- routers easily sort packets into their corresponding treatment classes by the TOS value, without having to know which flows or applications the packets belong to.

As work on diff-serv progresses, there has been a continued discussion on the second issue, which is whether differentiated services would need a signaling protocol for dynamic resource management. A commonly perceived notion is that the deployment of diff-serv would most likely start with manually configured resource allocations at network boundaries, however it remains an open issue how applications can achieve high quality services end to end.

We believe that end-to-end performance can be achieved through the concatenation of PHB's. We also believe that, although configurations can give us a jump start on deploying differentiated services, automatic protocol mechanisms will be needed in near future as the volume and scope of QoS-requiring traffic increase, to effectively and efficiently meet the ultimate goal of end-to-end QoS delivery.

2. A Picture of the Internet Today

The Internet today is made of the interconnection of multiple autonomous networks called autonomous systems, or administrative domains, because each is under a separate administrative control. Each domain contracts its neighboring domains to deliver the traffic; the neighbor domains, in turn, may pass the traffic to their neighbors, so on and so forth until packets reach their destinations. For example, a campus contracts an ISP to deliver its traffic and the ISP delivers the campus' traffic either over its own network if the destinations are connected to the same ISP, or otherwise pass the packets to other ISPs.

Following the AS-based network topology, as described above, today's Internet routing architecture takes a two-level hierarchical design. Each of the administrative domains, or Autonomous Systems, is free to choose whatever routing protocol deems proper to run internally. To assure global connectivity, neighbor domains speak BGP (Border Gateway Protocol) with each other to exchange network reachability information. Reachability information can be aggregated. For example, if nearby networks share common prefixes,

then reachability reports for them can be merged, so that a remote site will need to have one entry in its forwarding table showing the common prefix.

The separation of the Internal Gateway Protocol (IGP) and the Border Gateway Protocol (BGP), coupled with the ability to aggregate routing advertisements, provides the routing architecture with proven scaling characteristics.

3. A Framework for Scalable Resource Management

We propose a hierarchical approach to resource allocation for the global Internet. Following the development of the global routing architecture, we propose that individual administrative domains should be the basic control unit for resource management. Bilateral service agreements are made between neighboring administrative domains regarding the aggregate border-crossing traffic, while each administrative domain individually makes its own decision on strategies and protocols to use for internal resource management to meet internal clients QoS need and to fulfill external commitments.

We assume that a resource manager, named the Bandwidth Broker (BB) by Van Jacobson, exists in each administrative domain. A BB will be in charge of both the internal affairs and external relations regarding resource management and traffic control. Internally, a BB keeps track of QoS requests from users and applications, and allocates internal resources according to the domain specific resource usage policies which may specify which users can use how much resource or resource shares. The internal resource allocation can be done in a number of possible ways. For bandwidth-rich domains, perhaps little needs be done other than closely monitoring the network utilization level and re-provisioning accordingly. For bandwidth-poor domains or those with high variation in link capacities, the BB can make use of RSVP as the internal signaling protocol to reserve bandwidth for individual applications, as described in [RSVP].

Externally, a BB will be responsible for setting up and maintaining bilateral agreements with the BBs of the neighbor domains regarding the QoS handling of its border-crossing traffic flows. The BB for a transit domain (i.e. a provider network) must also keep those external service commitments to be within its internal resources capacity. The BB is a logical entity; actual implementations may take either a centralized or a distributed approach.

To scale with both the number of user data flows and the link speed, we propose that the bilateral agreements between BB's be in terms of differentiated traffic classes. The BB's instruct border routers of their own domains how much traffic each border router can export and import for each PHB class, so that border routers only need to perform QoS control for aggregate traffic classes.

This two-tier hierarchical framework gives us both the scalability in providing global scale QoS support, as well as the flexibility in managing resources within each administrative domain.

4. Summary

Today's Internet is made of interconnected autonomous networks controlled by administrative domains. Following the global routing architecture this position paper sketches out a two-tier hierarchical framework for global Internet resource management. More specifically, we propose to build a two-level hierarchy in resource management. To scale well, resource allocations between domains will be done for aggregate border-crossing traffic, while allocations within a domain can be done with a number of different possibilities depending on the user requirements and resource availability.

5. References

[RSVP] Y. Bernet, F. Baker, P. Ford, R. Yavatkar, L. Zhang, "A Framework for End-to-End QoS Combining RSVP/Intserv and Differentiated Services", Internet-Draft <draft-bernet-intdiff-00.txt>

7.3 Preliminary Results Evaluating Diffserv Services

Kathleen Nichols, Kedarnath Poduri, Juan-Antonio Ibanez
Bay Networks
knichols@baynetworks.com

Extended Abstract:
The differentiated services model for providing a quality of service in IP networks has evolved at a tremendous pace in the last year. Presently, concrete results or experiences with architectures built using the diffserv model have been few. Where such results have been reported, they have been quite rudimentary.

At Bay Networks, we have been working on both architectural aspects of diffserv and evaluation of the dynamics of diffserv traffic. Our own results are quite preliminary and, at this time, are simulation results only. Our simulations differ from others by including a TCP model rather than using abstract traffic loads, by using more complex topologies and traffic loads. We have focused on evaluating the behavior of two services built from diffserv primitives: Assured and Premium. Assured employs a drop preference behavior in each network node and Premium employs a behavior where the service rate at each network node must not be less than the arrival rate of packets within the same diffserv behavior aggregate. We will show results where the Premium service is implemented with two different mechanisms, both of which give the required behavior and for assured service using at least one, possibly two, drop preference mechanisms.

At this writing, we've performed Assured service simulations using a "RIO" (RED with In/Out, proposed by Dave Clark of MIT) dropper. Our traffic model has consisted largely of long-lived TCPs with some non-responsive and fixed rate sources. We found that the delivered rate to the user varies depending on RTT, topology, and the presence and type of other traffic. Thus, we feel that characterizing such a service with a requested rate would not lead to an appropriate user expectation or charging model. There is, however, clearly some preferential treatment experienced by the "in" traffic. It's not presently clear the best way to exploit or charge for that preferential treatment. We also performed some very preliminary simulations of the Premium service (as proposed by Van Jacobson of LBNL) using a simple priority mechanism at each network node and limiting and shaping the input traffic. These did give a service whose behavior was invariant under multiple levels of merging and, when implemented with a priority mechanism, showed jitter of less than one packet-time at the target rate. We plan to investigate the characteristics further and to compare them to implementing the service with a mechanism that is less stringent than simple priority, but still meets the requirements.

Our simulations are performed using the LBNL/UCB simulator ns-2. The TCP model is our own version, but is available through the "contributed code" portion of the ns-2 web page. Our traffic loads are a mixture of long-lived FTPs and simulated HTTP traffic, also available as "contributed code". The specific modules we've used to implement differentiated services in ns-2 are not currently publicly available, but are straightforward.

7.4 Designing for Public Networks

Bryan Lyles
Sprint Advanced Technology Labs

The Internet began its existence as an experimental, government funded, network offering a single class of service (best effort service) and operating in an environment where access was restricted and antisocial behavior was controlled by peer pressure. Today the Internet is a multibillion-dollar business, is designing an environment with multiple classes of service and has become a network open to all and where social controls are often ineffective. This evolution must lead it to become a *public network*. By "public network" we mean many things: legal and regulatory considerations, pricing of network services, importance to the national economy, ubiquity and the resulting total lack of control of the user base. Being a public network does not mean being a clone of the public telephone network, even though many of these considerations apply to that network – and first arose in that context, but it does imply concern about issues that have not traditionally been part of the Internet design criteria. We urge that public network issues be part of the design criteria for the Next Generation Internet.

The remainder of this position paper will briefly describe some examples of where the concerns of a public network might influence Internet design choices.

Consider a provider selling Internet access with a service level agreement (SLA) specified in terms of a packet loss rate. Now consider two customers A and B with T3 and T1 Internet access, respectively. If A transmits to B at T3 speeds, the network will be forced to drop the majority. Does the SLA still hold? How does the service provider determine whether the user is "obeying the rules?" What if the traffic is UDP traffic rather than TCP? These questions should be considered as part of any effort to redesign the Internet congestion feedback mechanisms.

What about the conditions under which we can charge for a service? Consider a source S and a receiver R using RSVP. Could S and R conspire to defraud their Internet providers? Suppose R sends a reservation towards S leaving behind a trail of reserved bandwidth. When the reservation reaches S it says "no" after a short wait. In the meantime R has received excellent service. Can the Internet provider(s) charge for the reserved network resources? This is an example of where technology meets the legal and regulatory systems. Under current rules the providers probably cannot charge for the bandwidth.

Federal law requires equal access to long-distance services and for the most part this has resulted in a competitive market for long-distance services. However, there is thriving business in Alternative Operator Services (AOS) at hotels and payphones. Some of these AOSs resell bulk-purchased long-distance at "unreasonable" markups to captive markets. Suppose that in the future you checked into a hotel providing Internet service to the rooms. If you made use of differentiated services could you choose the provider or would the hotel's default backbone provider be your only choice?

When we provide differentiated services we will price the service classes differently. We will also have to prove to our customers that they are receiving the service for which they are paying. We also need to be able to resolve customer problems when they cross carrier boundaries. We need measurements and measures. In order for two parties to agree on the significance of a measurement we need three careful definitions:

What is being measured?
How (and where) to measure it?
What are the acceptable readings?

The ability of measurements to isolate responsibility for an individual user needs to be part of the architecture.

Likewise, the architecture needs to comprehend issues related to economics and regulatory policy. For differentiated services, these considerations will impact (future) signaling mechanisms and whether we apply policy to individual flows or to aggregates.

For all services we need to assume that some individuals will attempt to cheat whenever pricing provides an incentive. (Differential services is a likely candidate for such pricing.) We need to design our protocols with theft of service in mind. Otherwise the Internet will end up as fraud-ridden as traditional analog cellular service.

These issues, and others, are aspects of designing for public networks. They imply that our simple models of technical elegance need to be augmented.

8. Acronyms

A

ALTQ	Alternate Queueing
ANL	Argonne National Laboratory
ARC	Ames Research Center
ARM	Application Response Measurement
ATM	Asynchonous Transfer Mode

B

BNL	Brookhaven National Laboratory

C

CBQ	Class-Based Queuing
CBR	Constant Bit Rate
CERES	Clouds and the Earth's Radiant Energy Sytem
Codecs	coder/decoders

D

DAAC	Distributed Active Archive Center
DARPA	Defense Advanced Research Projects Agency
DoE	Department of Energy
DRA	Distributed Routing Algorithm
DS3	Digital Signal 3 (44.7 Mbps)

E

Ebnet	EOSDIS Backbone Network
Esnet	Energy Sciences Network (DoE)
EDOS	EOS Data and Operations System
EOC	EOSDIS Operations Center
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ERDoS	End-to-End Resource Management for Distributed Systems
ESDIS	Earth Science Data and Information System

F

FIFO	First-In-First-Out

G

Gbps	Gigabits per second

H

HDTV	High Definition Television

I

IDL	Interface Description Language
IETF	Internet Engineering Task Force
IP (v4/v6)	Internet Protocol (versions 4 and 6)
ISP	Internet Service Provider
IST	Instrument Support Terminals
ITO	Information Technology Office (DARPA)

J

JSC	Johnson Space Center

K

KSC	Kennedy Space Center

L

LAN	Local Area Network
LANL	Los Alamos National Laboratory
LBNL	Lawrence Berkeley National Lab

M

MIB	Management Information Base
MJPEG	Motion Joint Photographic Experts Group
MODIS	Moderate Resolution Imaging Spectro-Radiometer
MPEG2	Moving Picture Experts Group Phase 2
MSFC	Marshall Space Flight Center

N

NAP	Network Access Point
NCAR	National Center for Atmospheric Research
NGI	Next Generation Internet
NISN	NASA's Integrated Services Network
NP	Network Performance
NREN	NASA Research and Education Network
nrtVBR	Non-RealTime Variable Bit Rate
NSI	NASA Science Internet
NSIDC	National Snow and Ice Data Center
NTSC	National Television Standards Committee

O

OC3	Optical Carrier 3 (155 megabits per second)
ORB	Object Request Broker
ORNL	Oak Ridge National Laboratory
OS	Operating System
OSPF	Open Shortest Path First

P

PHBs	Per-Hop Behaviors
PI	Principal Investigator
PIB	Path Information Base
PID	Process Identification Number
PNNI	Private Network to Network Interface
PVC	Permanent Virtual Circuit

Q

QA	Quality Assurance
QoS	Quality of Service

R

RSVP	Resource Reservation Protocol

S

SAAM	Server and Agent based Active Management
SCF	System Control Facility
SLA	Service Level Agreement
SLAC	Stanford Linear Accelerator Center
SNMP	Simple Network Management Protocol

T

TC	Traffic Conditioner
TCP	Transport Control Protocol
TTCP	Test Transport Control Protocol

U

UBR	Unspecified Bit Rate
UDP	User Datagram Protocol

V

VAFB	Vandenberg Air Force Base

W

WAN	Wide Area Network